Gould CONCEPT 32

32/67 CENTRAL PROCESSING UNIT

Technical Manual

April 1983

Publication Order Number: 303-000410-000

GOULD
Electronics

This manual is supplied without representation or warranty of any kind. Gould Inc., S.E.L. Computer Systems Division therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this publication or any material contained herein.

Copyright 1983
Gould Inc., S.E.L. Computer Systems Division
Printed in U.S.A.

HISTORY

The 32/67 Technical Manual, Publication Order Number **303-000410-000**, was printed April, 1983.

This manual contains the following pages:

Title page
Copyright page
iii/iv through xiv
1-1 through 1-10
2-1 through 2-96
3-1 through 3-93/3-94
4-1 through 4-77/4-78
IN-1 through IN-6

Digitized by the Internet Archive
in 2025

https://archive.org/details/manualsplus_06201

TABLE OF CONTENTS

Page

CHAPTER 1 GENERAL DESCRIPTION

1.1 Introduction 1-1
1.2 Central Processing Unit (CPU) 1-1
  1.2.1 Microsequencer (MS) Unit 1-1
  1.2.2 Instruction Execution (IE) Unit 1-1
  1.2.3 Cache SelBUS (CS) Unit 1-1
1.3 SelBUS 1-1
1.4 I/O Expansion Chassis 1-3
1.5 Input/Output Processor (IOP) 1-3
1.6 Integrated Memory Module (IMM) 1-3
1.7 Floating-point Accelerator (FPA) 1-3
1.8 Turnkey Panel and Operator Console 1-5
1.9 Internal Processor Unit (IPU) 1-5
1.10 Instruction Repertoire 1-5
1.11 Operating Mode 1-5
  1.11.1 Memory Environments 1-6
  1.11.2 Privileged State 1-6
  1.11.3 Addressing Option 1-7
1.12 Memory Management Hardware 1-7
1.13 Memory Write Protection 1-7
1.14 Memory Interleaving 1-8
1.15 I/O Transfers 1-8
1.16 Built-in Reliability and Maintainability Features 1-8
1.17 Software 1-8
1.18 Diagnostics 1-8
1.19 Specifications and Leading Particulars 1-9

CHAPTER 2 INITIALIZATION, OPERATION AND SOFTWARE CONTROL

2.1 Introduction 2-1
2.2 System Jumpers/Mode Switch 2-1
  2.2.1 Processor Select Switch 2-1
  2.2.2 Automatic Initial Program Load Jumpers 2-2
2.3 System Initialization 2-2
2.4 CPU/IPU Initialization 2-4
  2.4.1 Power-up Auto Restart 2-4
  2.4.2 Power-up Auto IPL 2-9
    2.4.2.1 Auto-IPL Limitations 2-9
    2.4.2.2 Auto-IPL Abort 2-10
  2.4.3 Power-up Automatic Trap Halt 2-10
  2.4.4 IPU Software Initialization 2-10
2.5 Manual IPL 2-11
  2.5.1 Reset 2-11
  2.5.2 Initial Program Load 2-25
    2.5.2.1 Firmware Reset 2-26
    2.5.2.2 IPL Device Address 2-26

2.5.2.3 IPL Identify I/O Protocol 2-32
2.5.2.4 IPL IOCDs 2-32
2.5.2.5 IPL Start I/O 2-35
2.5.2.6 IPL Data Transfers 2-35
2.5.2.7 IPL Termination 2-37
2.5.2.8 CPU Fill Pipeline 2-38
2.6 Powerdown Sequence 2-39
2.7 Internal Processor Unit (IPU) 2-42
2.7.1 IPU Operation 2-43
2.7.2 IPU Operational Characteristics 2-44
2.7.3 IPU Console 2-46
2.7.3.1 IPU Console Configuration 2-47
2.7.3.2 I/O Class Identification 2-49
2.8 Automatic Trap Halt 2-49
2.9 CPU/IPU Scratchpad 2-50
2.9.1 Scratchpad Image 2-50
2.9.2 Scratchpad Keyword 2-52
2.9.3 Device Entry 2-52
2.9.4 Interrupt Entry 2-52
2.10 Control of Features and Options 2-52
2.10.1 PROM or ACS Modes 2-55
2.10.1.1 PROM Mode 2-55
2.10.1.2 ACS Mode 2-55
2.10.1.3 ACS Mode Special Considerations 2-56
2.10.2 Development Support System/Diagnostic Processor Mode 2-57
2.10.3 Cache 2-57
2.10.4 Cache Control 2-58
2.10.4.1 CMC Instruction 2-59
2.10.4.2 SMC Instruction 2-59
2.10.4.3 IOP Panel Address Stop 2-60
2.10.4.4 Instruction Store (nonpresent memory) 2-60
2.10.4.5 Cache System Reset 2-60
2.10.5 Cache Memory Prefetch 2-60
2.10.6 Shared Memory 2-60
2.10.6.1 Interprocessor Bit Semaphores 2-61
2.10.6.2 Interprocessor Semaphore Considreations 2-64
2.10.6.3 SMC Read and Lock Control 2-65
2.10.6.4 SMC Shared Range Control 2-65
2.10.7 Writable Control Store (WCS) 2-67
2.10.8 Floating-point Accelerator (FPA) Option 2-68
2.11 Operator Indicators 2-70
2.11.1 Halt Indicator 2-70
2.11.2 Run Indicator 2-71
2.11.3 Wait Indicator 2-71
2.11.4 Interrupt Indicator 2-71
2.11.5 Parity Error Indicator 2-71
2.12 Operator Commands 2-72
2.12.1 Halt Command 2-72
2.12.2 Run Command 2-72
2.12.3 Reset Command 2-72
2.12.4 IPL Command 2-72
2.12.5 Attention Command 2-75
2.12.6 Write PSW Command 2-75
2.12.7 Write PC Command 2-75
2.12.8 Write PSD2 Command 2-75

2.12.9 Address Stop Commands 2-76
2.12.10 Address Stop Characteristics 2-77
2.12.11 Step Command..2-77
2.12.12 Memory Address Virtual Command 2-78
2.12.13 Read Effective Address Command...........................2-78
2.13 Instruction Attributes..2-79

CHAPTER 3 THEORY OF OPERATION

3.1 Introduction 3-1
3.2 Overview 3-1
3.2.1 MS Unit...3-1
3.2.1.1 Micro PC Address Selection Logic 3-1
3.2.1.2 Cache Data Out Logic 3-1
3.2.1.3 Scratchpad Logic...3-1
3.2.1.4 Control Store Logic 3-1
3.2.1.5 Order Structure 3-2
3.2.1.6 Turnkey Panel Interface....................................3-2
3.2.2 IE Unit 3-2
3.2.2.1 Instruction Pipeline..3-2
3.2.2.2 Set-up Logic...3-2
3.2.2.3 Execution Logic...3-2
3.2.2.4 Test Structure 3-2
3.2.2.5 Order Structure 3-2
3.2.3 CS Unit..3-2
3.2.3.1 SelBUS Interface Logic 3-2
3.2.3.2 Operand Effective/Instruction Logical Address Logic 3-13
3.2.3.3 Memory Management Logic...............................3-13
3.2.3.4 Cache Memory Logic 3-13
3.2.3.5 Machine Cycles 3-13
3.2.3.6 Clock Generation 3-13
3.3 Microsequencer (MS) Unit 3-13
3.3.1 Microsequencer Components.......................................3-15
3.3.1.1 Y Bus and DB Bus 3-15
3.3.1.2 Right Shifter (34) 3-15
3.3.1.3 Cache Data Out Registers (4, 35).......................3-15
3.3.1.4 I3 and I2 MS Copy Registers (27) 3-15
3.3.1.5 Instruction Decode (28, 29) 3-15
3.3.1.6 Decode Save (29)...3-16
3.3.1.7 Microinterrupt Control (31) 3-16
3.3.1.8 6:1 Input Address Selection Multiplexer (24, 25) 3-16
3.3.1.9 Microprogram Counter (uPC) (24, 25) 3-19
3.3.1.10 Microprogram Counter Save Register (26) 3-19
3.3.1.11 Microprogram Counter Latch (26) 3-19
3.3.1.12 Microprogram Counter Output Buffer (26) 3-19
3.3.1.13 WCS RAM Data Registers (2, 3, 4, 5)...................3-19
3.3.1.14 WCS Address Register (12) 3-19
3.3.1.15 N-Counter (36)...3-19
3.3.1.16 Control Store RAM (12-17) 3-20
3.3.1.17 Control Store PROM (18 - 21)..............................3-20
3.3.1.18 PROM & RAM Bank Enable Decode (11) 3-20
3.3.1.19 Order Structure (40) 3-20
3.3.1.20 Scratchpad (33) 3-20
3.3.1.21 Nibble Shifter (32)...3-20

3.3.1.22 Auto IPL (38) 3-20
3.3.1.23 Serial Input Poll (37) 3-21
3.3.1.24 ES and ED Field Control (39) 3-21
3.3.1.25 Turnkey Panel Interface (40) 3-21
3.3.2 Microsequencer Operation 3-21
3.3.2.1 Microprogram Counter 3-21
3.3.2.2 Microprogram Counter Control 3-22
3.3.2.3 Microinterrupt Control 3-22
3.3.2.4 Serial Interrupt Poll and IPU Trap 3-22
3.3.2.5 Data Flow 3-22
3.3.2.6 Micro PC Data Buffer 3-28
3.3.2.7 CROM Addressing 3-28
3.3.2.8 Control Store 3-28
3.3.2.9 Alterable Control Store (ACS) 3-29
3.3.2.10 Writable Control Store (WCS) 3-29
3.3.2.11 Order Structure 3-30
3.3.2.12 Turnkey Panel Interface Logic 3-31
3.3.2.13 Leading/Trailing Zeros Detect Logic 3-31
3.3.2.14 Decode Exceptions (Causes and Vectors) 3-31
3.4 Instruction/Execution (IE) Unit 3-32
3.4.1 IE-Unit Components 3-32
3.4.1.1 CAMUX Bus 3-32
3.4.1.2 Instruction Cache Register I3 (2) 3-32
3.4.1.3 Right Halfword Register (2) 3-32
3.4.1.4 I3 Right Halfword and RHW No-Op Detect Logic (2) 3-32
3.4.1.5 I2 Register (3) 3-32
3.4.1.6 I2 Indirect Register (4) 3-32
3.4.1.7 Predecoder (3) 3-32
3.4.1.8 I1/I0 Status Register (18) 3-32
3.4.1.9 I2 Bus 3-35
3.4.1.10 I1 Register (21) 3-35
3.4.1.11 Backdate Program Counter (14) 3-35
3.4.1.12 Multiplexer (5) 3-35
3.4.1.13 General Purpose Register (27, 28) 3-35
3.4.1.14 Base Register and Index Register (5, 6, 7) 3-35
3.4.1.15 Base Driver (7) 3-36
3.4.1.16 3-way Adder (8) 3-36
3.4.1.17 Effective Memory Address Register (9, 16) 3-36
3.4.1.18 Map Miss MAR (34) 3-36
3.4.1.19 LMAR Copy (4, 34) 3-36
3.4.1.20 32-Bit Constant (24) 3-36
3.4.1.21 Sign Extend (20) 3-36
3.4.1.22 Byte Constant (20) 3-36
3.4.1.23 IO Register (23) 3-37
3.4.1.24 I1/I0 Look Ahead Multiplexer (25, 26) 3-37
3.4.1.25 DB Bus Source Decode 3-37
3.4.1.26 Microprocessor 2901 (27, 28) 3-37
3.4.1.27 Y Bus 3-37
3.4.1.28 Y Bus Control (30) 3-37
3.4.1.29 Program Status Word Register (20) 3-37
3.4.1.30 Condition Code Logic (33) 3-37
3.4.1.31 CROM Bus 3-37
3.4.1.32 Test Structure (31) 3-37
3.4.1.33 Order Structure (32, 14) 3-38
3.4.1.34 Level Orders (34) 3-38

3.4.1.35 Control Register (CREG) (25, 26) 3-38
3.4.1.36 Clock Generator (34) 3-38
3.4.2 IE Unit Operation 3-38
3.4.2.1 CROM Bus Functions 3-38
3.4.2.2 Cache Multiplexer Bus Functions 3-43
3.4.2.3 MAR Polling Logic 3-43
3.4.2.4 Operand/Transaction Request Logic 3-44
3.4.2.5 Cache Transfer Type Logic 3-44
3.4.2.6 Microinterrupt Logic 3-44
3.4.2.7 I2 Bus Functions 3-49
3.4.2.8 DB Bus Functions 3-50
3.4.2.9 Y Bus Functions 3-50
3.4.3 Microprocessor Operation 3-50
3.4.3.1 Microinstruction Decode 3-51
3.4.3.2 Shifting 3-51
3.4.3.3 Random Access Memory (RAM) 3-53
3.4.3.4 Q-Register 3-53
3.4.3.5 Arithmetic Logic Unit (ALU) 3-53
3.4.3.6 Data Outputs 3-53
3.4.3.7 Stopclock Logic 3-54
3.5 Cache SelBUS (CS Unit) 3-56
3.5.1 Introduction 3-56
3.5.2 CS Unit Components 3-59
3.5.2.1 EMAR (9) 3-59
3.5.2.2 Macroprogram Counter (34) 3-60
3.5.2.3 Input Source Multiplexer (34) 3-60
3.5.2.4 I3 PC Register (31) 3-60
3.5.2.5 LMAR Counter (33) 3-60
3.5.2.6 MAR Comparator (13) 3-60
3.5.2.7 Map RAM (37) 3-60
3.5.2.8 Map Hit RAM (35) 3-60
3.5.2.9 Map Bypass Multiplexer (37) 3-60
3.5.2.10 Cache Memory (25, 26) 3-60
3.5.2.11 Cache Index RAMs (18, 19) 3-61
3.5.2.12 Cache Data In Register (19) 3-61
3.5.2.13 Left Shifters (11, 12) 3-61
3.5.2.14 Index MAR Buffers (14) 3-61
3.5.2.15 Cache Comparators (18, 19) 3-61
3.5.2.16 Cache Multiplexer (27, 28) 3-61
3.5.2.17 Shared Memory Detect Logic (6) 3-61
3.5.2.18 SelBUS Data Out Register (10) 3-61
3.5.2.19 Physical Memory Address Register (PMAR) (5) 3-61
3.5.2.20 Data Return Transfer (DRT) PMAR (6) 3-61
3.5.2.21 SelBUS Data In Register (2, 3) 3-62
3.5.2.22 SelBUS PMAR (4) 3-62
3.5.2.23 Map Data In Registers (2, 3) 3-62
3.5.2.24 PMAR Copy Register (8) 3-62
3.5.3 CS Unit Operation 3-62
3.5.3.1 Effective Address Logic 3-62
3.5.3.2 Memory Management Logic 3-63
3.5.3.3 Cache Memory Logic 3-64
3.5.3.4 SelBUS Interface 3-66
3.6 Instruction Pipeline 3-68
3.6.1 Backing Store Stage (I3) 3-68
3.6.2 Decode Stage (I2) 3-72

3.6.3 Vector Stage (I1) 3-73
3.6.4 Execute Stage (I0) 3-76
3.6.5 Advance Pipeline Routines 3-76
3.6.6 Pipeline Conflict Logic 3-79
3.6.7 Pipeline Violation Logic 3-79
3.6.8 Address Specification Errors 3-80
3.6.9 Auto Microinterrupt 3-80
3.7 Cache 3-80
3.7.1 Introduction 3-80
3.7.2 Primary Organization 3-83
3.7.3 Cache Addressing 3-83
3.7.4 Write to Cache 3-83
3.7.5 Least Recently Used (L.R.U.) Circuitry 3-84
3.7.6 Externally Originated Writes 3-84
3.7.7 CPU Write to Cache 3-84
3.7.8 Read From Cache 3-84
3.7.9 Cache Memory Control Register 3-85
3.7.10 Cache Shifters (F and C Bits) 3-85
3.7.11 Shared Memory Detection 3-86
3.7.12 Instruction/Operand Stop 3-86
3.8 Writable Control Store (WCS) 3-87
3.8.1 Entry and Use 3-87
3.8.2 WCS User Notes 3-87
3.9 Memory Map 3-88
3.9.1 Introduction 3-88
3.9.2 Memory Mapping Data Structures 3-88
3.9.2.1 Current Process Index 3-88
3.9.2.2 Master Process List 3-88
3.9.2.3 Map Image Descriptor List 3-91
3.9.2.4 Map Image Descriptor 3-91
3.9.2.5 Map Initialization 3-91
3.9.3 Map Entries 3-92
3.9.4 Map Load 3-92
3.9.5 Look-aside Buffer 3-92
3.9.6 Map Bypass Multiplexing 3-93
3.9.7 Logical to Physical Address Conversion 3-93

CHAPTER 4 MICROPROGRAMMING

4.1 The Microword 4-1
4.2 Firmware Responsibility 4-1
4.3 Machine Cycles 4-2
4.4 Microword Field Descriptions 4-4
4.4.1 Sequence Field 4-4
4.4.2 Test Field 4-6
4.4.3 Register A Field 4-6
4.4.4 Register B Field 4-7
4.4.5 External Source Field 4-7
4.4.6 Internal Source Field 4-7
4.4.7 ALU Function Field 4-7
4.4.8 Internal Destination Field 4-8
4.4.9 External Destination Field 4-8
4.4.10 Order Enable and Order Group Fields 4-8
4.4.11 P, C, and H Fields 4-9
4.4.11.1 L Branch Address Field 4-9

4.4.11.2 Branch Address Field 4-9
4.4.11.3 Leap Address Field 4-9
4.4.11.4 Hop Address Field 4-9
4.4.11.5 Literal/Scratch Address field 4-9
4.4.11.6 Byte Constant Field 4-9
4.4.11.7 CC Select Field 4-9
4.4.11.8 L-Order Select Field 4-9
4.4.11.9 ALU Shift Codes Field 4-10
4.4.11.10 SelBUS Codes Field 4-10

INDEX IN-1

LIST OF ILLUSTRATIONS

1-1 Gould CONCEPT 32/67 Computer Nucleus 1-2
1-2 I/O Expansion Chassis 1-4
1-3 Turnkey Panel 1-6

2-1 Auto-IPL Jumpers, Example 2-3
2-2 CPU/IPU Power-up Sequencing Flowchart (3 sheets) 2-5
2-3 System Reset Flowchart (11 sheets) 2-14
2-4 Initial Program Load Flowchart (5 sheets) 2-27
2-5 MPX-32 System Distribution Tape Format 2-36
2-6 Diagnostic Tape/Disc Distribution Facility Format 2-37
2-7 Powerdown Sequencing Flowchart 2-40
2-8 Allocation of CPU/IPU Scratchpad 2-51
2-9 CPU/IPU Device Entry Format 2-53
2-10 CPU/IPU Interrupt Entry Format 2-54
2-11 CPU/IPU Shared Memory 2-61
2-12 Remote Shared Memory Configuration 2-62

3-1 32/67 CPU Functional Diagram 3-3
3-2 32/67 CPU Block Diagram 3-5
3-3 MS Unit Functional Diagram 3-7
3-4 IE Unit Functional Diagram 3-9
3-5 CS Unit Functional Diagram 3-11
3-6 Basic Timing Diagram and Clock Nomenclature 3-14
3-7 MS Unit Block Diagram 3-17
3-8 Microprogram Counter Block Diagram 3-23
3-9 Macroinstruction to Microinstruction Block Diagram 3-25
3-10 IE Unit Block Diagram 3-33
3-11 IE Unit Test Structure 3-40
3-12 Microword Test Structure 3-41
3-13 Four-bit Slice Microprocessor Block Diagram 3-52
3-14 CS Unit Block Diagram 3-57
3-15 Instruction Pipeline Block Diagram 3-69
3-16 2901 Register Selection 3-77
3-17 Cache Block Diagram 3-81
3-18 Memory Management Hardware Block Diagram 3-89
3-19 Mapping Data Structures 3-90

4-1 Microword 4-3
4-2 CROM Cycle and CREG Cycle Descriptions 4-4

LIST OF TABLES

1-1 Specifications and Leading Particulars 1-10

2-1 Powerup Automatic Trap Halt Indications and Causes 2-11
2-2 Auto-IPL and IPL Fault Isolation 2-12
2-3 Class F IPL Memory Contents 2-33
2-4 Class E IPL Memory Contents 2-34
2-5 Class F First IPL Record Format 2-34
2-6 CPU Read Status Word Format 2-42
2-7 Power Up/Down Scratchpad Memory Image Location Redefinition 2-43
2-8 IPU and CPU I/O Classes 2-45
2-9 IPU and CPU Trap Vector Locations 2-45
2-10 Multiprocessor Non Read and Lock SBM, Example 2-64
2-11 Multiprocessor Use of Non Read and Lock Instruction to Modify Semaphore, Example 2-66
2-12 Alphabetical List of System Control Panel Commands 2-73
2-13 State Indicator Pairs and Listing 2-74
2-14 Instruction Attributes (15 sheets) 2-80
2-15 Address Specification Rules (2 sheets) 2-95

3-1 Condition Code Select 3-42
3-2 IE Unit Cache Transactions 3-45
3-3 Microinterrupt Priority Levels 3-47
3-4 ALU Function Control 3-54
3-5 IE Unit Stopclock Conditions 3-55
3-6 I3 Instruction Attribute Decode 3-71
3-7 I2 Instruction Attribute Decode 3-74

4-1 Sequence Field CROM 00 through 03 (2 sheets) 4-10
4-2 Test Field CROM 04 through 11 4-12
4-3 Test Group 0 (3 sheets) 4-13
4-4 Test Group 1 (3 sheets) 4-16
4-5 Test Group 2 (3 sheets) 4-19
4-6 Test Group 3 (2 sheets) 4-22
4-7 Register A-Field CROM 12 through 15 (2 sheets) 4-24
4-8 Register B-Field CROM 16 through 19 (2 sheets) 4-26
4-9 External Source Field CROM 20 through 23 (5 sheets) 4-28
4-10 External Source Overlay CROM 20 through 23 (3 sheets) 4-33
4-11 Internal Source Field CROM 24 through 27 4-36
4-12 ALU Function Field CROM 28 through 31 4-37
4-13 Internal Destination Field CROM 32 through 35 4-38
4-14 External Destination Field CROM 36 through 39 (3 sheets) 4-39
4-15 External Destination Overlay CROM 36 through 39 (3 sheets) 4-42
4-16 Order Enable Field and Order Group Fields CROM 40 through 51 4-45
4-17 Order Group 0 (5 sheets) 4-46
4-18 Order Group 1 (5 sheets) 4-51
4-19 Order Group 2 (6 sheets) 4-56
4-20 Order Group 3 (4 sheets) 4-62
4-21 Condition Code Select Field CROM 52 through 55 4-66

4-22 Level Order Select Field CROM 52 through 55 (3 sheets) 4-67
4-23 ALU Shift Code Field CROM 54 and 55 (2 sheets) 4-70
4-24 SelBUS Codes Field CROM 56 through 59 (3 sheets) 4-72
4-25 SelBUS Overlay CROM 56 through 59 (3 sheets) 4-75

CHAPTER 1

GENERAL DESCRIPTION

1.1 Introduction

This chapter provides a brief overall description of the Gould CONCEPT 32/67 Computer Nucleus (figure 1-1). The main elements of the Gould 32/67 computer nucleus include the three slot CPU (Microsequencer (MS) Unit, Instruction Execution (IE) Unit, and Cache SelBUS (CS) Unit), the Model 8001 Input/Output Processor (IOP) with an IOP related device interface board (IOP-DI), and one Model 3028 Integrated Memory Module (IMM). The optional Floating-point Accelerator (FPA) Model 3611 is also shown in the illustration.

1.2 Central Processing Unit (CPU)

The 32-bit Gould CONCEPT 32/67 CPU is based on a three board design (MS, IE, and CS Units). The basic configuration provides the capability of alterable control store (ACS) and writable control store (WCS). ACS enables modification, under software control, of the basic microprogram elements of the system. WCS provides the functional capability of allowing user written microprograms to be loaded and executed under software control. These user programs support applications such as the scientific accelerator.

1.2.1 Microsequencer (MS) Unit

The MS unit contains the microprogram in programmable read only memory (PROM). It processes operations such as microinstruction decode, microinterrupt control, increment, jump and branch. The ACS and WCS functions are implemented on the MS unit.

1.2.2 Instruction Execution (IE) Unit

The IE unit contains the macroinstruction pipeline and the 2901 microprocessor. The unit performs effective address calculations, operand prefetching, and instruction execution steps.

1.2.3 Cache SelBUS (CS) Unit

The CS unit contains a fast access memory, a memory map, and the SelBUS interface. The cache enables portions of the main memory to be accessed at high speed. The memory map transforms logical addresses into references to actual locations in main memory or cache. The SelBUS interface monitors the SelBUS for externally originated memory reads or writes and establishes priority for I/O transfers.

1.3 SelBUS

The SelBUS is a bidirectional and time-divisional multiplexed bus that provides signals, power, and ground to the chassis. SelBUS slots not used for the full complement of IMMs are available for regional processing units, input/output (I/O) processors, or high-speed data interfaces. The SelBUS contains 184 parallel lines which include a 32-bit bidirectional data bus, a 24-bit bidirectional destination bus, and other lines for control signals, priority, power, and ground. All SelBUS lines operate synchronously at a 150-nanosecond clock rate.

The 32/67 CPU uses the SelBUS as a high-speed data path to transfer data and commands between the CPU and either

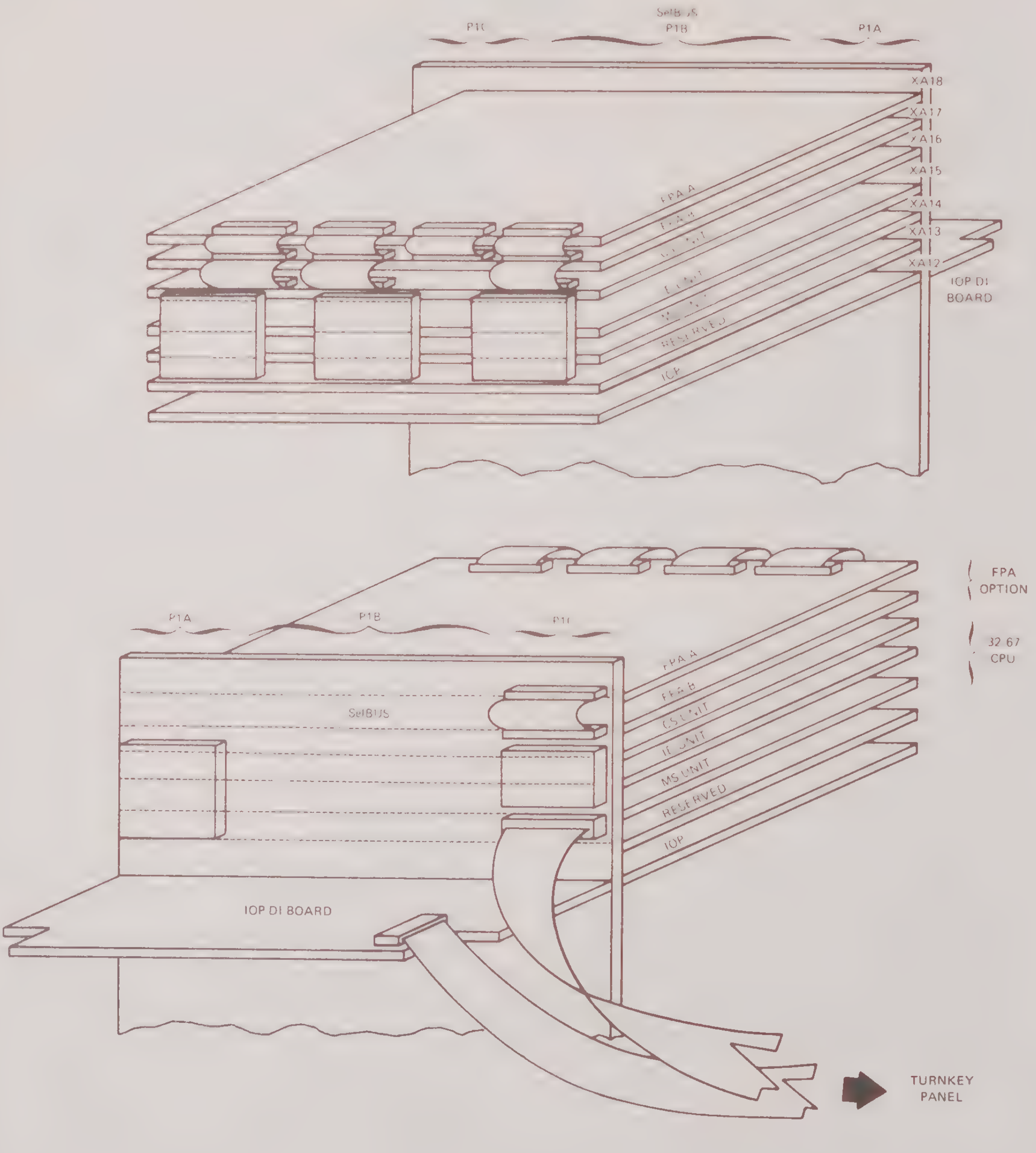

SelBUS
P1C
P1B
P1A
XA18
XA17
XA16
XA15
XA14
XA13
XA12
FPA A
FPA B
CS UNIT
IE UNIT
MS UNIT
RESERVED
IOP
IOP DI BOARD
P1A
P1B
P1C
FPA OPTION
32/67 CPU
FPA A
FPA B
CS UNIT
IE UNIT
MS UNIT
RESERVED
IOP
SelBUS
IOP DI BOARD
TURNKEY PANEL
830450

Figure 1-1. Gould CONCEPT 32/67 Computer Nucleus

the main memory by an IMM, any of the input/output processors (namely, I/O channels, I/O controllers), or other SelBUS-connected modules such as a real-time option module (RTOM).

1.4 I/O Expansion Chassis

The chassis provides SelBUS expansion and I/O capabilities. A 50-pin flat cable connects the 32/67 chassis to the I/O expansion chassis MP Bus via the IOP. The I/O expansion chassis (see figure 1-2) adds eight bus slots to a basic system. The expansion chassis is split into two electrically separated sections which permit one IOP to control all eight device controllers or allow two IOPs to control four device controllers each.

The MP bus (see figure 1-2) is a medium-speed asynchronous bus designed for the transfer of I/O data. Up to 8 single-slot I/O controllers can be installed for connection to the MP Bus.

1.5 Input/Output Processor (IOP)

The IOP is an input/output multiplexing channel that also utilizes a bipolar bit slice MP2901 microprocessor. It can support 124 subchannels with a maximum bus transfer rate of 384K words per second. The microprocessor 16-bit data structure produces 32-bit transfers on the SelBUS whenever possible. The IOP communicates with the 32/67 CPU and main memory over the SelBUS.

The IOP consists of the following interrelated elements: a SelBUS interface, a MP Bus interface, the IOP proper, and system control panel (SCP) device dependent logic. The SelBUS interface provides the communications path between the IOP and the 32/67 CPU, or the IOP and memory. The IOP proper has a control memory that contains the microprogram (firmware) for controlling the SelBUS and IOP interfaces. The IOP circuits consist of control logic for operating the MP Bus, the SCP interface, and the receiver/drivers necessary to communicate with the MP Bus controllers.

The IOP combines the functions of a real-time option module (RTOM), the SCP, and the operator console device, which all share the same SelBUS interface and microprocessor. The controllers of the multiplexing channels are implemented on 12.5 by 15 inch boards. These boards plug into the I/O expansion chassis.

1.6 Integrated Memory Module (IMM)

The IMM is a high density, memory system contained on a single 15-inch by 18-inch printed circuit board. Each IMM board consists of 256K word (1MB) of metal oxide semiconductor (MOS) memory, onboard refresh logic, SelBUS interface logic, data formatter/error correcting logic, and associated drivers. With expansion chassis options, memory capacity may be increased to a maximum of 4M words (16M bytes).

IMMs operate with a 600-nanosecond read access time and a 300-nanosecond write cycle time. By using fully overlapped read cycles, the average read cycle time reduces to 300 nanoseconds. Also, in respect to the 32/67 CPU operation, the IMM is capable of two reads in 900 nanoseconds. The IMM refresh cycle time is 300 nanoseconds, occuring every 15 microseconds. A two-deep input buffer is available for all cycles except halfword and byte write cycles; therefore, the write halfword/byte cycle time is 600 nanoseconds. IMM storage modules may be, two-way, or four-way interleaved.

The MOS memory is organized into 39-bit words: 32 data bits plus 7 error correction bits which are associated with each word in memory and correct all single-bit errors. Double-bit errors are detected and reported.

1.7 Floating-point Accelerator (FPA)

The optional Model 3611 Floating-point Accelerator (FPA) performs floating-point addition, subtraction, multiplication, and

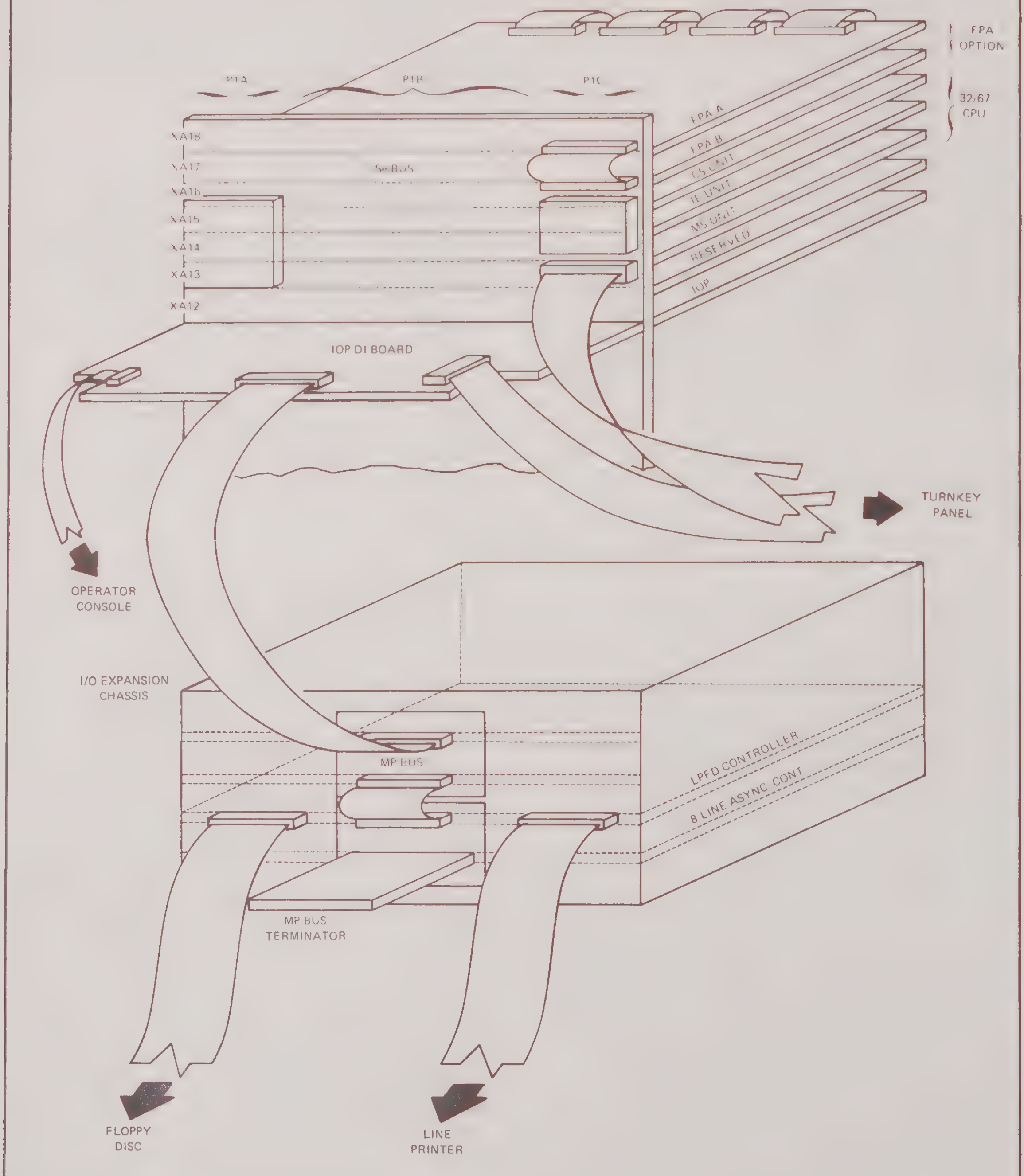

FPA
OPTION
P1A
P1B
P1C
32/67
CPU
XA18
XA17
SelBUS
XA16
XA15
XA14
XA13
XA12
FPA A
FPA B
CS UNIT
IE UNIT
MS UNIT
RESERVED
IOP
IOP DI BOARD
TURNKEY
PANEL
OPERATOR
CONSOLE
I/O EXPANSION
CHASSIS
MP BUS
LPFD CONTROLLER
8 LINE ASYNC CONT
MP BUS
TERMINATOR
FLOPPY
DISC
LINE
PRINTER
830451

Figure 1-2. I/O Expansion Chassis

division on single precision (32 bits) and double precision (64 bits) operands. The FPA also performs fixed-point multiplication. The FPA is contained on two standard size logic boards which connect directly to the SelBUS (see figure 1-1). In a CONCEPT/32 chassis, the FPA is dedicated to SelBUS slots A17 and A18 in the logic/memory chassis.

1.8 Turnkey Panel and Operator Console

Figure 1-3 shows the turnkey panel that is standard with the system. The panel allows user interaction with the processor system. The panel includes the master POWER ON/OFF switch, four functional LED indicators (RUN, WAIT, HALT, INTERRUPT) for both CPU and IPU, system REBOOT pushbutton switch, MODE selector switch, and a PROCESSOR SELECT switch.

The operator console is an alphanumeric CRT used to access the CPU, via the IOP, for system operation or domestic commands. It also provides a display of messages required to operate and maintain the operating system. The character set contains a total of 128 displayable characters (96 ASCII characters plus 32 control characters). It is capable of displaying 24 lines of information with a maximum of 80 characters per line.

1.9 Internal Processing Unit (IPU)

The 32/6780 contains an IPU. An IPU consists of a second set of CPU boards. This second processor is configured as an IPU by the turnkey panel PROCESSOR SELECT switch and processor identify jumpers on the two CS units. The IPU provides the facility to offload tasks from the CPU to the IPU, thereby increasing the computational performance of the system. At the hardware level, the two processors are identical; they share the same SelBUS and memory. The IPU may be provided with panel capabilities and console I/O capabilities by the optional IPU Console. This option comprises an IOP, a CRT, and cabling.

1.10 Instruction Repertoire

The CPU instruction set includes fixed-point and floating-point arithmetic instructions. The instruction set provides the computational and data-handling capabilities required for widely differing applications. All instructions are fully detailed in the 32/67 Reference Manual. The instruction set includes:

- Register-to-register operations with a 16-bit instruction format to improve program execution time.
- Fixed-point integer and floating-point arithmetic operations in single (32-bit) and double (64-bit) precision formats.
- Complete selection of logical operations (AND, OR, exclusive OR) for bytes, halfwords, words, and doublewords.
- Comparison operations for bit, byte, halfword, word, and doubleword operands.
- Supervisior call instruction that allows a program access to specified operating system services.
- Shift operations (left and right) of word or doubleword; including logical, circular, and arithmetic shifts.
- Immediate operand instructions for greater storage efficiency and increased speed.

1.11 Operating Mode

The 32/67 CPU operates in the program status doubleword (PSD) mode. Individual bits of the PSD, when set, regulate the memory environment, the privilege state, and the addressing option. The PSD mode, whether the CPU is operating in the privileged or unprivileged state, creates the environment required to run the mapped programming executive (MPX-32).

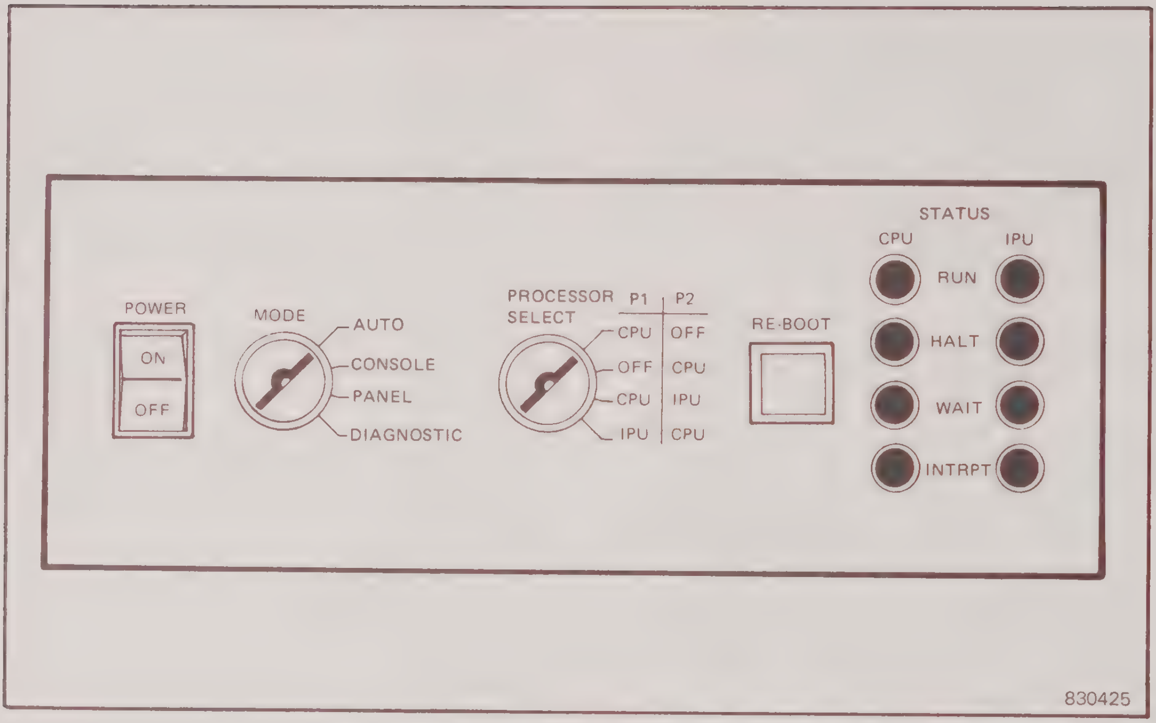


Figure 1-3. Turnkey Panel

1.11.1 Memory Environments

The 32/67 CPU may operate in a mapped environment or an unmapped environment, which is contingent upon operating conditions and applications of the Gould CONCEPT 32/67 Computer Nucleus.

Under each of these environments, the user controls the selection of the nonextended or extended addressing options, and these options determine the rules for logical address generation.

The memory environment is controlled by the software operating system which also determines the rules for transposing logical addresses into physical addresses.

In nonbase mode, the logical address space beyond the first 128K words may be used for operands only, and the upper limit of this space depends on whether the CPU is operating in the mapped or unmapped environment. In base mode, the logical address space may be anywhere in memory.

For the mapped environment, the upper limit of the extended space is 4M words, and the distribution of the logical address space within the physical address spaced is controlled by the memory map.

For the unmapped environment, the upper limit of the extended space is 4M words, and all logical addresses are equal to physical addresses. Indexed addressing is necessary to achieve addressing above 128K words. In each memory reference instruction, bits 9 and 10 designate one of three general-purpose registers to be used as an index register.

Indirect addressing facilitates table linkages and permits keeping data sections of a program separate from procedure sections for ease of maintenance.

1.11.2 Privileged State

If the privileged state bit in the program status doubleword is set, privileged instructions can be executed. If the bit is reset, any attempt to execute a privileged instruction will cause a privileged violation trap.

The privileged state of operation is a qualifying condition that permits only the operating system to function. In this state, the CPU performs the input/output instructions and all of its control functions to monitor any part of the system. This is because the CPU, when operating in the privileged state, bypasses the map write protect logic so that any portion of memory can be accessed.

In the mapped environment with the privileged bit not set, memory protection is in effect, and all privileged operations are prohibited. The unprivileged state of operation indicates that user access to memory is permissible. The user may access the operating system's portion of memory, but the user cannot write in any of that space.

1.11.3 Addressing Option

The memory address generation (base or nonbase) is controlled by the CPU memory environment (mapped or unmapped). Once generated by the CPU, the address calculation is further modified by the memory addressing option currently in effect.

In nonbase mode, the memory addressing options are either extended or nonextended. The nonextended addressing option allows the CPU to access instructions or operands in the first 128K words of memory. The extended addressing option provides the access to any bit, byte, halfword, word, or doubleword residing anywhere in memory up to 4M words of logical address space in a mapped environment, and up to 4M words of logical address space in an unmapped environment. A 19-bit address field is provided in all memory reference instructions for memory addressing. In base mode, programs may reside anywhere in memory.

1.12 Memory Management Hardware

The memory management hardware is a logical tool that organizes the individual map blocks of a user's task. The memory management hardware uses a memory map (random access memory) to transform logical space (addresses) into physical space (addresses).

When the user's task is loaded into memory, it is dispersed into noncontiguous map blocks throughout physical memory. All of the map blocks used for a specific user's task are considered the physical (real) space of that task. Physical blocks of memory can be common to many logical address spaces. Thus, multiple users may have access to some of the same physical address space and share those common blocks of memory.

The memory management hardware permits full utilization of all available memory by allowing user programs to be loaded into, and executed from, anywhere in physical memory.

1.13 Memory Write Protection

The memory protection system provides write protection for individual memory map blocks. However, write protection is overridden when the CPU is operating in the privileged state.

When the CPU is operating in the mapped environment, and using either the nonextended or extended option, each 2K-word map block of logical program address space can be write protected for any quarter (512W) of any map block. This is done by the setting of the appropriate protect/unprotect bit in the map image descriptor for that particular map block.

The memory write protection feature protects each user from other unprivileged users and prevents inadvertent destruction of information in protected memory areas. This feature also can be used to protect the integrity of programs essential to critical real-time applications.

No protection is provided in the unmapped environment; direct access to all available physical address space is attainable, through extended indexing or base mode.

1.14 Memory Interleaving

Memory interleaving is a built-in hardware feature that distributes sequential addresses into independently operating memory modules. Interleaving increases the probability that a processor can gain access to a given memory location without encountering interference from other processors. Thus, interleaving reduces memory access time and increases the throughput rate.

When a system consists of two memory modules (or a multiple thereof), memory can be two-way interleaved. If a system has four modules (or a multiple thereof), memory can be four-way interleaved. With two-way interleaving, even addresses are assigned to the second and fourth memory modules and odd addresses to the first and third memory modules. Four-way interleaving assigns every fourth address to its respective memory module.

1.15 I/O Transfers

The 32/67 CPU supports class 3, E, D, and F I/O controllers and devices. There is, in fact, no support provided in the software operating system for class E operation. However, the software does support class D operation (MPX operating system terminology). Class E and D may be considered the same, with the exception that with class D the I/O device can access up to 16M bytes of main memory.

1.16 Built-in Reliability and Maintainability Features

There is error correction in MOS memory for all memory read accesses.

Address stop features permit the operator or maintenance personnel to stop on any instruction address, stop on any memory read reference address, or stop on any memory write reference address.

Automatic traps (for error or fault conditions) have masking capability and facilitate recoverability under program control.

CPU traps, which provide for detection of a variety of CPU and system fault conditions, are designed to facilitate system recoverability.

1.17 Software

Operating systems are the mapped programming executive MPX-32 (at level MPX 3.2 and greater) and UNIX. The software language processors include: FORTRAN66+ and FORTRAN77+, ANSI COBOL, BASIC, assembler, utilities, and applications software for real-time and scientific users. To accommodate the 32/67 CPU map implementation, the software supports the following:

- 2K word map block
- 2048 map blocks per user
- Write protection contained in the map

NOTE

UNIX is a trademark of
Bell Laboratories.

1.18 Diagnostics

The 32/67 CPU operates with the following nucleus diagnostics:

CPU Part 1 (CN.CP1.A)

CPU Part 2 (CN.CP2.A)

CPU Part 3 (CN.CP3.A)

Base Register (CN.BRD.A)

Trap (CN.TRP.A)

Interrupt (CN.INT.A)

Memory Management (MAP) (CN.MMM.A)

Effective Address (F & C bit) (CN.EAD.A)

DEXP (CN.DXP.A)
Floating Point (firmware), floating-point accelerator, and fixed to float/float to fixed (CN.FPT.A)

Interval Timer (CN.ITD.A)

IOP Console (CN.CON.A)

IPU Trap (CN.IPT.A)

Control Store (CN.CSD.A)

Multiport Memory (CN.MPM.A)

Cache Memory (CN.CMD.A)

1.19 Specifications and Leading Particulars

Specifications and leading particulars are provided in table 1-1.

Table 1-1
Specifications and Leading Particulars

Word Length	32 bit (4 bytes)
Data Sizes	1, 8, 16, 32, 64 bits
General-Purpose Registers	8 (3 of which can be used for indexing in non base, or 7 in base register mode)
Base Registers	8
Floating-point Arithmetic (firmware)	Integral to processor (standard)
Floating-point Accelerator (firmware)	Optional
Logic Board	
Dimensions	15 by 18 inches (each)
SelBUS continuous throughput	6.67M words/second (26.67M bytes/second)
Power	
Voltage	+5 volts (+5%, -10%)
Current IE unit	19.5 amperes
CS unit	19.0 amperes
MS unit	22.0 amperes
Environmental	
Operating	
Temperature (Celsius)	+10 degrees to +40 degrees
Relative humidity	5% to 95%
Storage	
Temperature (Celsius)	-40 degrees to +60 degrees
Relative humidity	2% to 95%
System Integrity features	Memory write protection Power fail safe Memory error correction circuitry (ECC) Arithmetic exception Privilege violation Nonpresent memory
Memory Management Scheme	
MAP	2K x 16-bit
MAP hit RAM	128 x 16-bit
MAP page	2048 word granularity

CHAPTER 2

INITIALIZATION, OPERATION, AND SOFTWARE CONTROL

2.1 Introduction

This chapter describes system initialization and operation. Full details on the software control of features and options are provided.

2.2 System Jumpers/Mode Switch

There are several jumper types in a 32/67 computer system which must be properly set before the system is initialized. Since system modules may vary for different system models and configurations, special-purpose jumpers for individual system modules are not provided here. The user must consult the 32/67 system manual for individual system modules and configurations.

The 32/67 CPU does not affect any of the SelBUS transfer priority generation or recognition jumpers. The CPU may be directly installed into a SelBUS backplane without changing any SelBUS transfer priority jumpers, assuming the jumpers have been previously set for standard SelBUS communication rules.

The CPU operates at SelBUS priority 0 for SelBUS I/O transfers and at pseudo priority 23 for SelBUS memory transfers. Pseudo priority 23 results from the absence of any other SelBUS device polling on SelBUS priorities 0 through 22. With both CPU and IPU installed, the CPU will operate at pseudo priority 23 and the IPU will operate at priority 22. When an IPU is installed, SelBUS 22 must be enabled (SelBUS terminator jumper removed).

2.2.1 Processor Select Switch

The 32/67 has an IPU option; the IPU (internal processing unit) consists of a second processor that plugs into the SelBUS and shares the SelBUS and memory with the CPU. The IPU differs from the CPU in that the IPU has limited I/O capability. The IPU provides the CPU with the capability of off-loading a task to the IPU while the CPU starts a new task or resumes a task in progress.

When a CPU and IPU are installed, or when a CPU is identified as processor 2, the SelBUS priority 22 jumper must be removed (enabled) on the SelBUS terminator.

In a system with two CPUs, a distinction must be made between which operates as processor 1 and which operates as processor 2. This is accomplished by a jumper on the CS unit which designates one of the two CPUs as processor 2. At system reset, the processor select switch is examined to determine processor identity (CPU, IPU, or OFF LINE).

Processor 1 operates at pseudo priority 23 which is the lowest priority on the SelBUS. When operating at priority 23 the processor is not required to perform a full SelBUS poll cycle and only needs to determine if another SelBUS unit is polling. This limited poll enables the processor to execute a one clock memory write and a four clock memory read operation after a cache miss.

Processor 2 operates at priority 22 and is required to perform a full SelBUS poll. A full poll causes the processor to execute a two clock memory write and a five clock memory read after a cache miss.

2.2.2 Automatic Initial Program Load Jumpers

The 32/67 CPU provides an automatic initial program load (auto-IPL) feature that provides the CPU with the capability of initiating an initial program load (IPL) program (software bootstrap program) during power-up initialization with memory contents that do not meet the requirements of the power-up automatic restart (auto-restart) feature. Firmware disables the processor designated as the IPU from initiating an auto-IPL sequence.

The 16 auto-IPL jumpers are used to enable the auto-IPL feature and to provide a 15-bit SelBUS physical address and subaddress of a dedicated I/O device that contains the software bootstrap program. Typically, this I/O device is the system disc containing the operating system software. The auto-IPL device must power up on line.

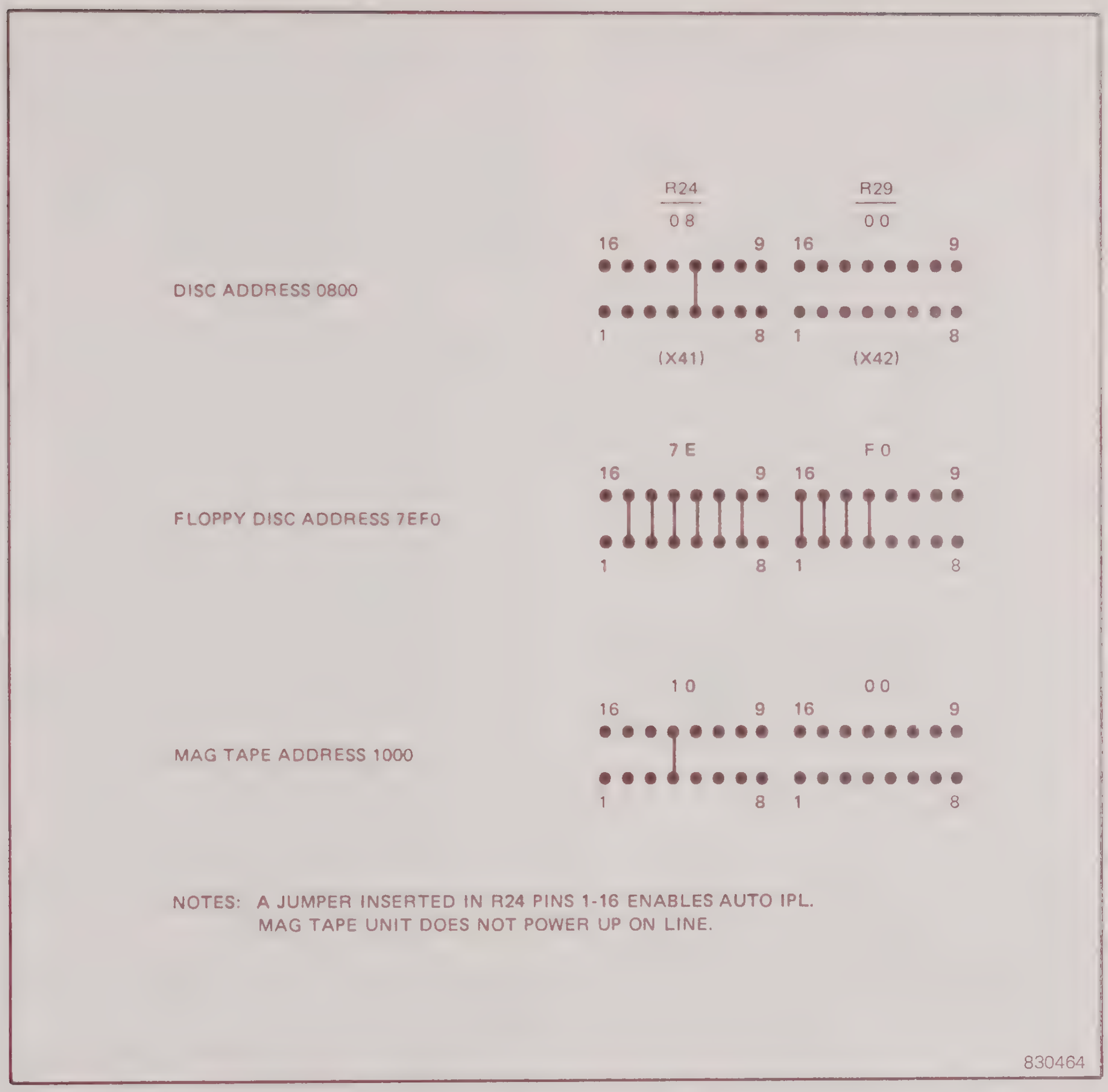
Figure 2-1 illustrates the position and function of the auto-IPL jumpers on the MS board. The enable/disable jumper is a single bit. The I/O device physical address field of the jumpers is a 7-bit field, and the jumpers selected must match the SelBUS physical address of the dedicated I/O device. The I/O device subaddress field is an 8-bit field that provides the subaddress of the dedicated device. The subaddress selected in this field must match the subaddressing rules of the corresponding I/O device.

Each jumper inserted in the auto-IPL jumper group provides a logical one for functional operation.

2.3 System Initialization

System initialization is a complex procedure requiring both hardware and software participation. System initialization includes the following functions:

1. System reset is executed by the hardware or the operator.
2. IPL of the software bootstrap program is initiated by the hardware or the operator.
3. The CPU scratchpad I/O and interrupt entries are loaded and the required base addresses in the scratchpad initialized by the software bootstrap program.
4. The dedicated memory interrupt and trap vector locations are initialized by the software bootstrap program.
5. Rollout of the CPU scratchpad image to dedicated memory locations 300 through 6FC occurs, and if power-up auto restart is to be used, the scratchpad keyword is assured to be in the memory image. This rollout function is performed by the software bootstrap program.
6. Software enables CPU traps.
7. Software initializes all previously unused memory words to purge parity errors or uncorrectable data errors.
8. If shared memory is present (other than CPU-IPU shared memory), software must describe the shared memory boundaries (upper boundary and lower boundary) by executing the shared memory control (SMC) instruction. This is defined in 128K-word blocks.
9. If an IPU is present and on line during initialization, software must enable the read and lock function in the CPU by executing the SMC instruction.
10. If an IPU is present, software in the IPU must:
    - a. Initialize the scratchpad.
    - b. Execute a SMC to describe shared memory boundaries if present.
    - c. Execute a SMC to enable read and lock.


R24
08
R29
00
16
9
16
9
DISC ADDRESS 0800
1
8
1
8
(X41)
(X42)
7E
F0
FLOPPY DISC ADDRESS 7EF0
10
00
MAG TAPE ADDRESS 1000
830464

NOTES: A JUMPER INSERTED IN R24 PINS 1-16 ENABLES AUTO IPL.
MAG TAPE UNIT DOES NOT POWER UP ON LINE.

Figure 2-1. Auto-IPL Jumpers, Example

11. In the CPU and IPU, software must initialize ACS by copying PROM to ACS or loading a new control store image into ACS then updating the control store image in ACS by loading applicable firmware patches.

12. In the CPU and IPU, with the WCS present, software must load the WCS control store image into WCS.

13. In the CPU and IPU, software must enable the ACS mode of operation by executing a SET CPU instruction to enable WCS.

14. In order to retain software loaded scratchpad, ACS mode, and WCS contents in both CPU and IPU, software must ensure that the CPU keyword is loaded into the CPU and that the IPU keyword is loaded into the IPU.

In the preceding list, steps 1 through 3 can be referred to as CPU initialization, and steps 4 through 14 can be referred to as software initialization.

Overall system initialization is executed in one of two methods as follows:

1. During system power-up sequences, where the memory contents are unreliable or designed to inhibit power-up auto-restart functions, auto IPL must also be enabled.

2. By manual usage of the reset and IPL functions of the system panel.

2.4 CPU/IPU Initialization

Power-up sequencing occurs whenever AC power is applied to the system and the power-up and reset signals are gated to the SelBUS. Power-up sequencing may cause one of the following three actions:

1. Power-up auto restart
2. Power-up auto-IPL
3. Power-up automatic trap halt

Figure 2-2 provides a flowchart of the CPU/IPU power-up sequencing. Refer to Control of Features and Options later in this chapter for definitions of PROM, ACS, and ROMSIM modes

2.4.1 Power-up Auto Restart

Power-up auto restart is a standard feature of the 32/67 CPU that is designed to provide the capability of restarting or continuing a software program that was interrupted by a power outage. Note that the power-up auto-restart feature does not guarantee the integrity of any I/O operation that was in progress when the power outage occurred or the physical positioning of the I/O media following a power outage.

For correct usage of the power-up auto-restart feature, the user should also be familiar with the power-down trap and the information provided in the power-down trap context block. Obviously, a successful power-down trap must precede any power-up trap. The power-down trap sequence is provided and described later in this chapter. During the power-down sequence, firmware does not roll out the CPU and IPU scratchpad keywords, but does roll out CPU and IPU status words and configuration words.

For the successful execution of a power-up auto-restart trap, the following operating conditions must be met:

1. The CPU/IPU and system must have been previously intialized with software traps enabled.

2. Software must maintain the CPU scratchpad image in the dedicated memory scratchpad locations 300 through 6FC.

3. The scratchpad image must contain the CPU and IPU scratchpad keyword.

4. A power-down trap must have been executed.

5. The integrity of memory must be preserved during the power outage (battery backup).

Both CPU and IPU execute similar auto restart sequences. However, the IPU does not roll in a scratchpad image and does not attempt an auto-IPL if the IPU keyword is incorrect in the memory scratchpad image.

If any of the preceding conditions are not met, the power-up auto restart is aborted and either an auto IPL or an automatic trap halt is executed. If the above conditions are met, the CPU/IPU tests for a pending power-down condition. Therefore, if another power-down is pending, the CPU/IPU will hang in an internal loop, waiting for the power to fail, thus preventing recursive power-down traps. If the pending power-down signal is false, the CPU/IPU transfers control to the power-up trap software via the power-up trap context block.

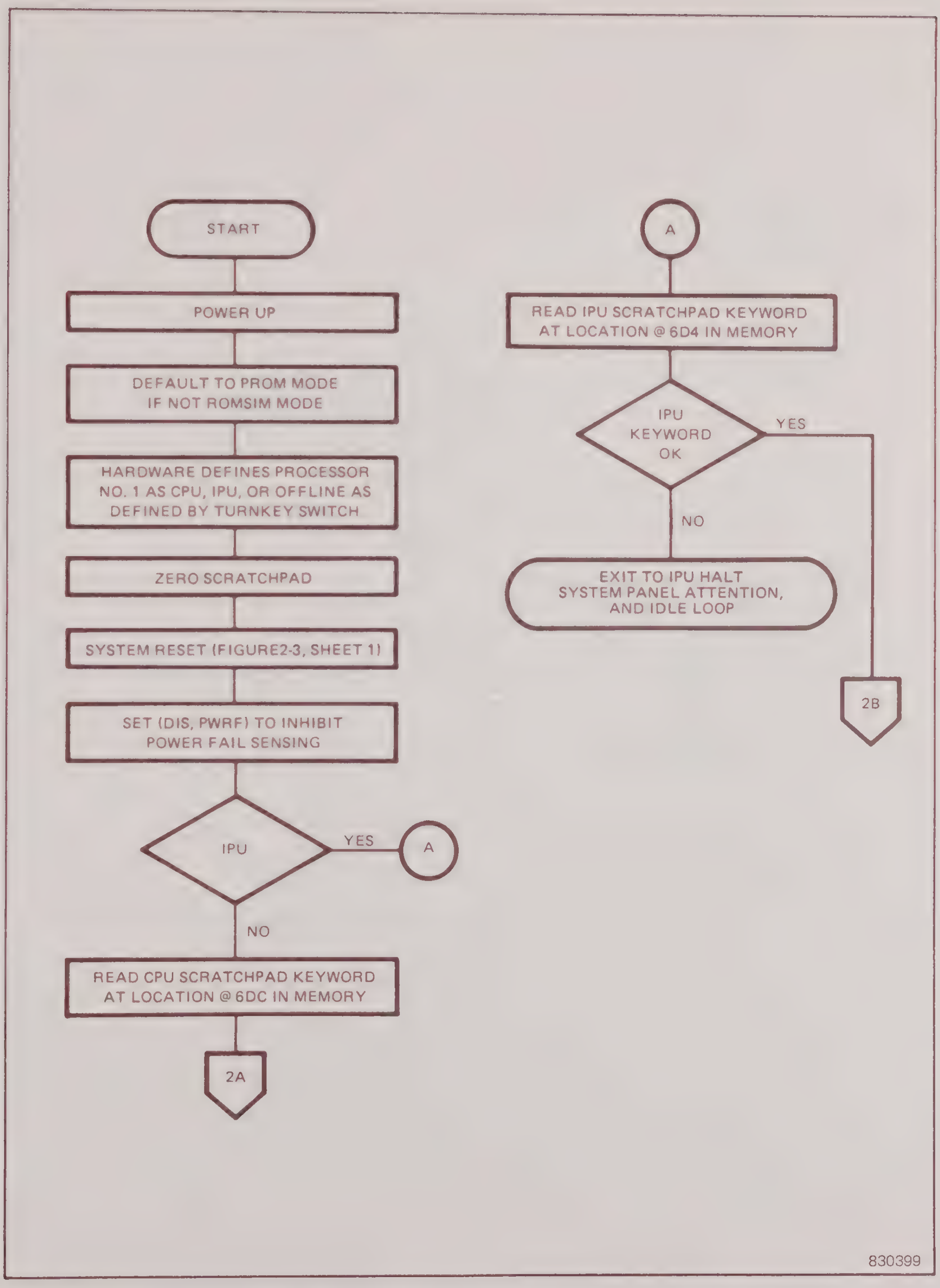

START
POWER UP
DEFAULT TO PROM MODE IF NOT ROMSIM MODE
HARDWARE DEFINES PROCESSOR NO. 1 AS CPU, IPU, OR OFFLINE AS DEFINED BY TURNKEY SWITCH
ZERO SCRATCHPAD
SYSTEM RESET (FIGURE2-3, SHEET 1)
SET (DIS, PWRF) TO INHIBIT POWER FAIL SENSING
IPU
YES
A
NO
READ CPU SCRATCHPAD KEYWORD AT LOCATION @ 6DC IN MEMORY
2A
A
READ IPU SCRATCHPAD KEYWORD AT LOCATION @ 6D4 IN MEMORY
IPU KEYWORD OK
YES
NO
EXIT TO IPU HALT SYSTEM PANEL ATTENTION, AND IDLE LOOP
2B
830399

Figure 2-2. CPU/IPU Power-up Sequencing Flowchart (Sheet 1 of 3)

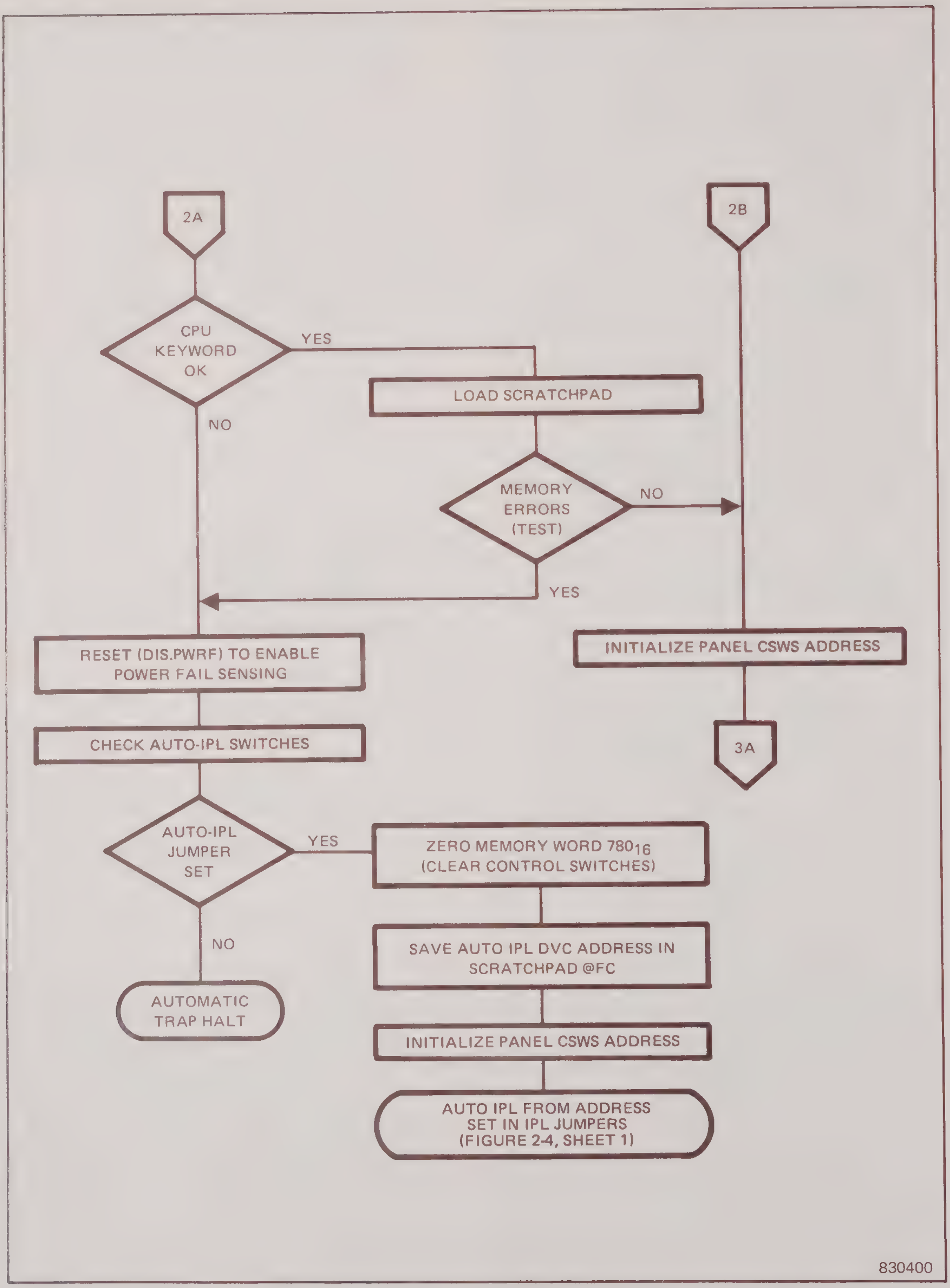

2A
2B
CPU KEYWORD OK
YES
NO
LOAD SCRATCHPAD
MEMORY ERRORS (TEST)
NO
YES
RESET (DIS.PWRF) TO ENABLE POWER FAIL SENSING
INITIALIZE PANEL CSWS ADDRESS
3A
CHECK AUTO-IPL SWITCHES
AUTO-IPL JUMPER SET
YES
ZERO MEMORY WORD 78_{16} (CLEAR CONTROL SWITCHES)
NO
SAVE AUTO IPL DVC ADDRESS IN SCRATCHPAD @FC
AUTOMATIC TRAP HALT
INITIALIZE PANEL CSWS ADDRESS
AUTO IPL FROM ADDRESS SET IN IPL JUMPERS (FIGURE 2-4, SHEET 1)
830400

Figure 2-2. CPU/IPU Power-up Sequencing Flowchart (Sheet 2 of 3)

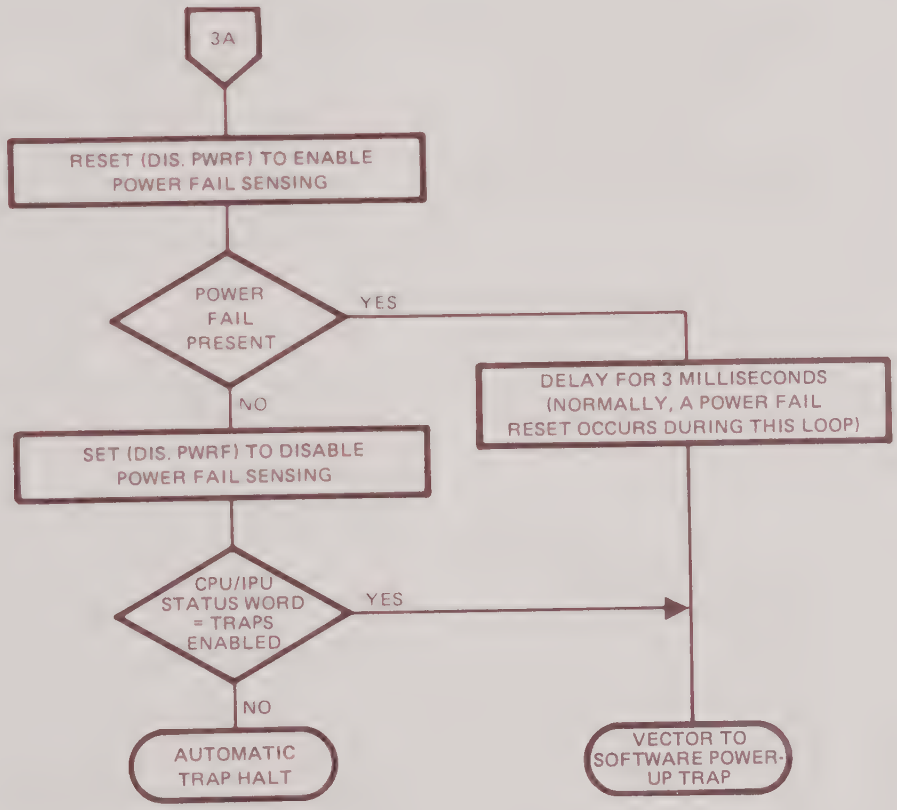

3A
RESET (DIS. PWRF) TO ENABLE POWER FAIL SENSING
POWER FAIL PRESENT
YES
DELAY FOR 3 MILLISECONDS (NORMALLY, A POWER FAIL RESET OCCURS DURING THIS LOOP)
NO
SET (DIS. PWRF) TO DISABLE POWER FAIL SENSING
CPU/IPU STATUS WORD = TRAPS ENABLED
YES
NO
AUTOMATIC TRAP HALT
VECTOR TO SOFTWARE POWER-UP TRAP
830401

Figure 2-2. CPU/IPU Power-up Sequencing Flowchart (Sheet 3 of 3)

When the software power-up trap handler receives control, both the power-up and power-down trap detection hardware has been disabled by CPU hardware. The trap detection hardware will remain disabled until software executes either a load program status doubleword (LPSD) instruction or a load program status doubleword and change map (LPSDCM) instruction.

Since the pending power-down signal was tested prior to the transfer of control to the power-up trap software, the software in the power-up trap handler can operate for a maximum of two milliseconds without enabling the power-up and power-down trap detection hardware. During this time period, power-up and power-down traps are disabled.

Software should make the power-down/power-up trap handler reentrant. Once reentrance has been achieved, and within two milliseconds, software should reenable the power-up and power-down detection logic by executing either a LPSD or LPSDCM instruction.

A restart of the power-down interrupted software can be achieved by executing a LPSDCM instruction through the old program status doubleword (PSD) stored in the power-down trap context block.

Auto restart can be inhibited by zeroing the CPU/IPU scratchpad keywords in the memory scratchpad image.

When the software power-up trap handlers receive control of both CPU and IPU, the operating features of the CPU and IPU have been set to their default values by the power-up and system reset firmware. Specifically, the state of the CPU is:

1. The scratchpad memory image has been loaded into the CPU scratchpad.
2. All cache banks are enabled.
3. The CPU is operating in the PROM mode and the content of the ACS is unknown.
4. All WCS, if present, has been initialized to 'long branch to undefined instruction'.
5. Hardware FPA, if present, is enabled.
6. Traps are enabled
7. Read and lock is disabled.
8. Shared memory detection is disabled and the boundary registers are clear.
9. The CPU internal status word and configuration word reflect the current operating state of the CPU and not the states that were in effect when power down occurred.
10. Power up/down sensing is disabled.
11. PSD1 and PSD2 reflect the PSD1 and PSD2 provided by the CPU power-up trap context block.

The power-up trap state of the IPU is as follows:

1. The IPU scratchpad base addresses (TCB, ICB, and MPL) are set to default values.
2. All remaining scratchpad in the IPU is clear.
3. All cache banks are enabled.
4. The IPU is operating in the PROM mode and the contents of the ACS is unknown.
5. All WCS, if present, has been initialized to 'long branch to undefined instruction'.
6. Hardware FPA, if present, is enabled.
7. Traps are enabled.
8. Read and lock is disabled.

9. Shared memory detection is disabled and the boundary registers are clear.

10. The IPU internal status word and configuration word reflect the current operating state of the IPU and not the states that were in effect when power down occurred.

11. PSD1 and PSD2 reflect the PSD1 and PSD2 provided by the IPU power-up trap context block.

To restore the operating states of the CPU and IPU, software must use the status words and configuration words that were stored in the memory scratchpad image by the power-down sequence. Specifically, the IPU scratchpad must be reloaded and then ACS, WCS, read and lock, and shared memory must be restored in both the CPU and IPU.

The IPU power-up software will receive control approximately 500 microseconds before the CPU power-up software. During this time, the IPU software should not attempt to signal the CPU via the SIPU instruction since the corresponding CPU trap will be cleared by the CPU power-up and reset sequence that is still in progress. The recommended IPU power-up software sequence is as follows.

1. Reload the IPU scratchpad.

2. Initialize read and lock and shared memory boundaries.

3. Save interrupted GPRs and base registers on a stack (make power-up software reentrant).

4. Enable power fail sensing by executing a LPSD or LPSDCM.

5. Wait for a SIPU from the CPU.

6. Continue initialization of ACS and WCS using CPU resources.

7. Reload base registers and GPRs.

8. Return to software interrupted by power down, or allow the CPU to dispatch IPU as required.

2.4.2 Power-up Auto IPL

Power-up auto IPL may occur when the conditions for a power-up auto restart cannot be met. Specifically, the conditions that cause an attempt at auto IPL are:

1. The memory scratchpad image contains parity errors or uncorrectable data errors.

2. The memory scratchpad image does not contain the CPU scratchpad keyword.

If either of the above conditions are true, the CPU power-up sequence examines the 16 auto-IPL jumpers located on the MS board. If the most significant jumper is true, indicating that auto-IPL is enabled, the CPU initiates an IPL-I/O sequence to the SelBUS physical address and sub-address provided by the remaining 15 jumpers. If the auto-IPL feature is not enabled, or if a detectable error occurs within the IPL-I/O sequence, an automatic trap halt is executed. The specific functionality of the IPL-I/O sequences is described later in this chapter.

During auto IPL, the CPU clears memory location 780. This prevents false detection of memory errors when initialiation software accesses the software control switches (CSWS). Additionally, scratchpad location @FC is set to the auto IPL device address to flag auto IPL restarts. During manual IPL sequences, scratchpad location @FC is set to zero.

The IPU does not execute an auto-IPL. It enters the idle loop and waits for the CPU (software) to initialize it.

2.4.2.1 Auto-IPL Limitations

For the auto-IPL feature to operate correctly the IPL-I/O device must power up in an on-line condition and be fully operable. I/O devices that do not power up on-line, or devices that do not correctly position the I/O media during IPL, cannot be used with the auto-IPL feature.

I/O devices that will power up on-line, but that indicate inoperable status prior to achieving on-line, will not work with the auto-IPL feature. However, I/O devices that indicate "busy" status prior to achieving on-line will work with the auto-IPL.

In general, the phase II disc processor, the input/output processor (IOP) disc processor, and the IOP line printer floppy disc controller function correctly with the auto-IPL feature. Magnetic tape processors, however, do not function with the auto-IPL.

2.4.2.2 Auto-IPL Abort

During an auto-IPL sequence, the CPU is executing a tight firmware loop that monitors the IPL sequence and waits for the IPL to terminate. In some cases, the IPL may not terminate, which causes the CPU to remain in the IPL sequence. Should this occur, the operator must abort the IPL sequence using the following procedure:

1. Use the IOP console and enter the panel mode (@@P keyboard sequence).
2. Issue a halt command. Ordinarily, the CPU will halt; otherwise, an operator error message is generated.
3. Ignore the operator error message, if present, and issue a reset (RST) command. The CPU should respond with a normal reset display. If the response is operator error, the reset sequence should be repeated.
4. Proceed to manual IPL.

2.4.3 Power-up Automatic Trap Halt

In power-on sequences, where auto restart and auto-IPL cannot be executed, the CPU will execute an automatic trap halt.

The automatic trap halt is characterized by a halt display on the system control panel with the interrupt active indicator turned on. The program counter portion of the displayed PSD word 1 (PSD1) indicates the memory address of the trap vector that caused the automatic trap halt. Tables 2-1 and 2-2 list the various causes and malfunctions of automatic trap halts in the power-up sequence. A more complete discussion of automatic trap halt is provided later in this chapter.

2.4.4 IPU Software Initialization

The operating system views the IPU and CPU as two totally symmetrical processors relative to their options and features. The following listing describes the procedures that must occur to initialize the IPU and the software checks that should occur to ensure that the IPU and CPU are symmetrical processors. For the IPU initialization to occur properly, the IPU software should have a copy of the current CPU status word and configuration word. All CPU trap vector locations (TVLs) must be initialized by CPU software prior to initializing the IPU (IPU traps are always enabled).

1. IPU read and lock must be enabled.
2. The IPU shared memory boundaries should be set as designated in the CPU configuration word.
3. The CPU memory scratchpad image should be loaded into the IPU scratchpad. Then, IPU scratchpad location F0 hex must be set to the correct IPU trap table base address.
4. If software requires the IPU to retain its scratchpad contents and option/feature configuration through system resets, the software must load the IPU keyword into the IPU scratchpad at location F7.

Table 2-1
Powerup Automatic Trap Halt Indications and Causes

Display PSD Word 1	Trap Type	Cause
8000 0084 or 8000 0086	Power up	Scratchpad image had memory errors and auto IPL was not enabled.
8000 0084 or 8000 0086	Power up	Scratchpad image was okay. Auto restart was attempted, but traps were disabled at the time of the power-down trap. Check the CPU status word in the memory scratchpad image.
8000 009C or 8000 009E	Machine check trap	Auto-IPL or IPL-I/O sequence error. (See Table 2-2.)

5. If software requires the IPU to perform power-up auto restart, the software must place the IPU scratchpad keyword in the CPU memory scratchpad image at location 6D4.

6. If the CPU is operating in the PROM mode, the IPU should be placed in the PROM mode.

7. If the CPU is operating in the ACS mode, the IPU ACS must be loaded and enabled.

8. If the CPU is operating with the hardware FPA disabled, the IPU FPA should be disabled.

9. If the CPU FPA is enabled, but the IPU FPA is either nonpresent or cannot be enabled, a nonsymmetrical configuration error is present. In this case, the system operator should be notified or, alternatively, the CPU FPA can be disabled.

10. The IPU WCS, if present, must be loaded to match the CPU WCS.

11. If the IPU WCS configuration is different from the CPU WCS configuration, a nonsymmetrical configuration error exists and the system operator should be notified.

2.5 Manual IPL

Manual IPL is initiated from the console by using the system reset and IPL functions. The specific operations required to execute reset and IPL are described in the console description later in this chapter. The following description cover the CPU and system reaction to the system reset and IPL functions.

2.5.1 Reset

The system reset function can only be executed while the CPU is in a halt mode and the halt indicator is on. This particular restriction is enforced by the IOP and not the CPU. Similarly, the IPU and IPU-IOP can initiate the manual system reset if the IPU is halted. If either the CPU or IPU initiates the system reset, both processors are reset.

Table 2-2
Auto-IPL and IPL Fault Isolation

NOTE

The following secondary indications are valid when a machine trap automatic trap halt occurs during IPL or auto IPL.

Secondary Indication	Cause
Memory locations 0 and 4 contain data errors or nonspecific data.	Addressed IPL channel was not present or inoperable.
Memory locations 0 and 4 contain @0200 7FFF and @0000 0000.	Addressed IPL channel was not class F, and no IPL data was transferred from device to memory.
Memory locations 0 and 4 contain @0200 0000 and @6000 0078.	Addressed IPL channel was class F, but no IPL data was transferred from the device to memory.
The IPL channel is not class F and memory location 0 contains a valid macro instruction (normally a branch instruction). The branch instruction also looks like a valid PSD word 1.	The IPL has worked and data has been transferred from the device to memory. The bootstrap software should be memory resident. Insure that the automatic trap halt did not occur as a result of the bootstrap software.
The IPL channel is class F and memory locations 0 and 4 contain a valid PSD. PSD word 1 also looks like a branch instruction.	IPL data has been transferred from the device to memory. IPL-I/O termination has not occurred which indicates a malfunction in the CPU, I/O channel, or the I/O device.
The IPL channel is class F and memory locations 0 and 4 contain a class F status doubleword. Word 2 of the doubleword indicates channel, subchannel, or device errors.	Class F channel, subchannel, or I/O device errors. Also, I/O media flaws.

When the CPU/IPU receives the reset function, a hardware master clear signal is generated within the CPU/IPU and a reset signal is driven to the SelBUS causing all circuit cards that plug-in to the SelBUS to be reset or master cleared. The CPU/IPU master clear signal causes the micro PC to be reset, thus reinitializing the firmware. The master clear signal also clears all hardware flags and registers.

As part of the hardware reset sequence, the hardware examines the PROCESSOR SELECT turnkey switch. The hardware then defines processor 1 as CPU, IPU, or OFF LINE and processor 2 as CPU, IPU, or OFF LINE as defined by the turnkey switch.

When the hardware master clear is complete, the firmware reinitialization takes place to restore the CPU to an operational condition, but in a halted mode. Figure 2-3 provides a flowchart of the firmware reinitialization or system reset flow.

An overview of the software apparent reset condition is as follows:

1. The CPU and IPU enter the PROM mode.
2. The software general-purpose registers (GPR) are set to zeros.
3. The scratchpad (SPAD) device entry and interrupt entry dynamic flags (bits 0 through 3) are set to zero in scratchpad locations 00 through EF.
4. The cache is cleared.
5. If the CPU or IPU keyword is not present in the corresponding scratchpad location F7, the following default parameters are set.
    a. The trap table base address is set to 0000 0080 in scratchpad location F0 (0020 trap table base address in the IPU).
    b. The interrupt table base address is set to 0000 0100 in scatchpad location F1.
    c. The input/output command doubleword (IOCD) table base address is set to 0000 0700 in location F2.
    d. The master process list (MPL) base address is set to 0000 0788 in scratchpad location F3.
    e. The CPU or IPU scratchpad keyword in location F7 is set to zero.
    f. The CPU or IPU status word in scratchpad location F9 is set to zero, disabling CPU/IPU traps.
    g. In the IPU only, IPU status word bit 26 (enable traps) is set in scratchpad location F9, enabling only IPU traps.
    h. The CPU and IPU shared memory registers are cleared, disabling shared memory and read and lock functions.
    i. Both banks of WCS, if present, are initialized to 'long branch to undefined instruction' and the appropriate WCS bits are set or cleared in the configuration word to indicate WCS configuration.
    j. All cache banks are turned on, and the configuration word cache indicator bits (27 thru 31) are set.
    k. If present, the hardware FPA is enabled and the corresponding bit in the CPU/IPU status word is set or cleared to reflect the FPA condition.

NOTE

If the scratchpad keyword is present, the table base addresses are not altered.

6. If the CPU or IPU keyword is present in the corresponding scratchpad location F7, the following parameters are not changed.
    a. The scratchpad table base address (TVL, IVL, MPL, and IOCDL).
    b. CPU or IPU keyword.
    c. CPU or IPU status word.
    d. Shared memory registers.
    e. The state of read and lock.
    f. The contents of WCS, if present.

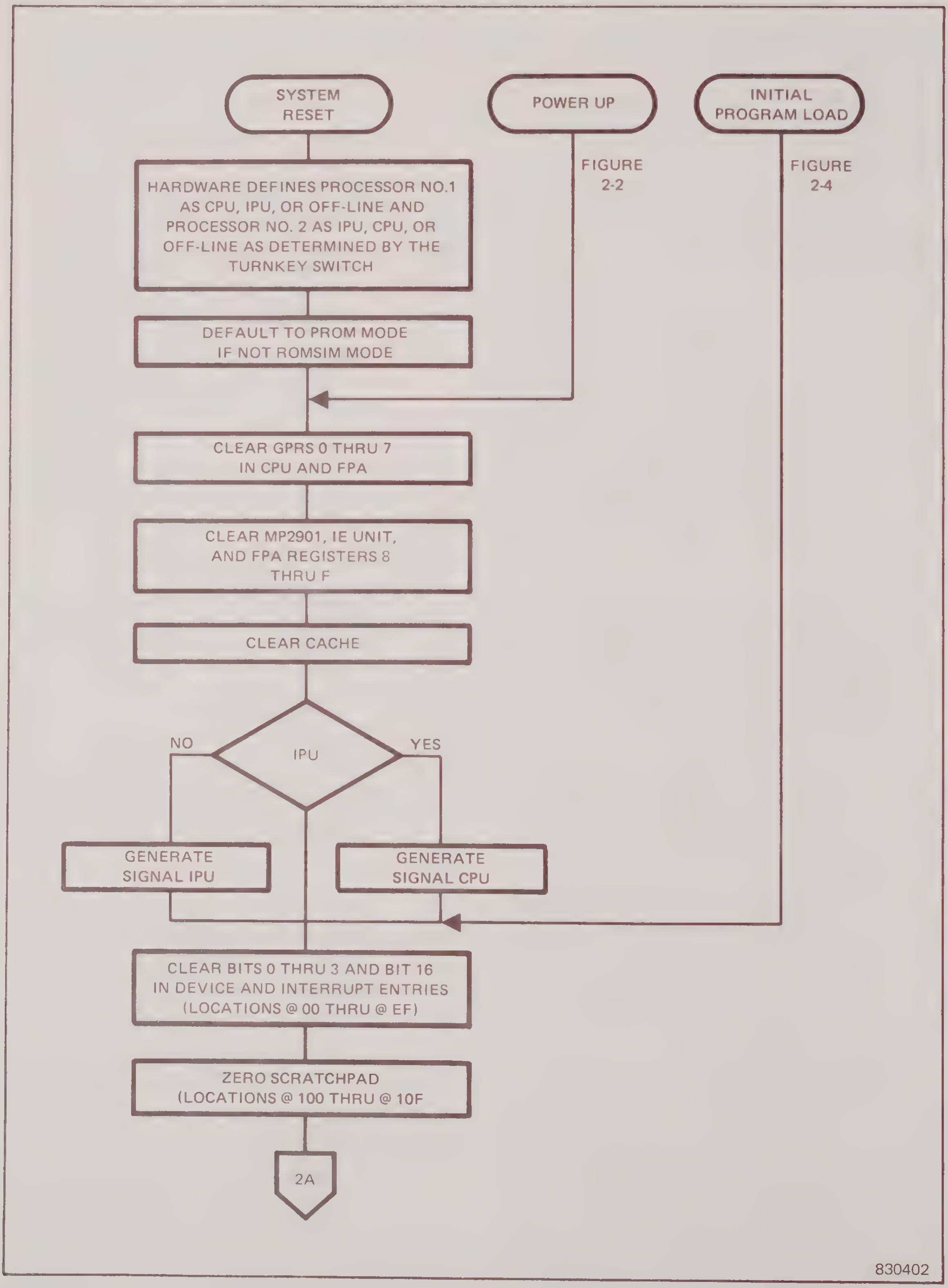
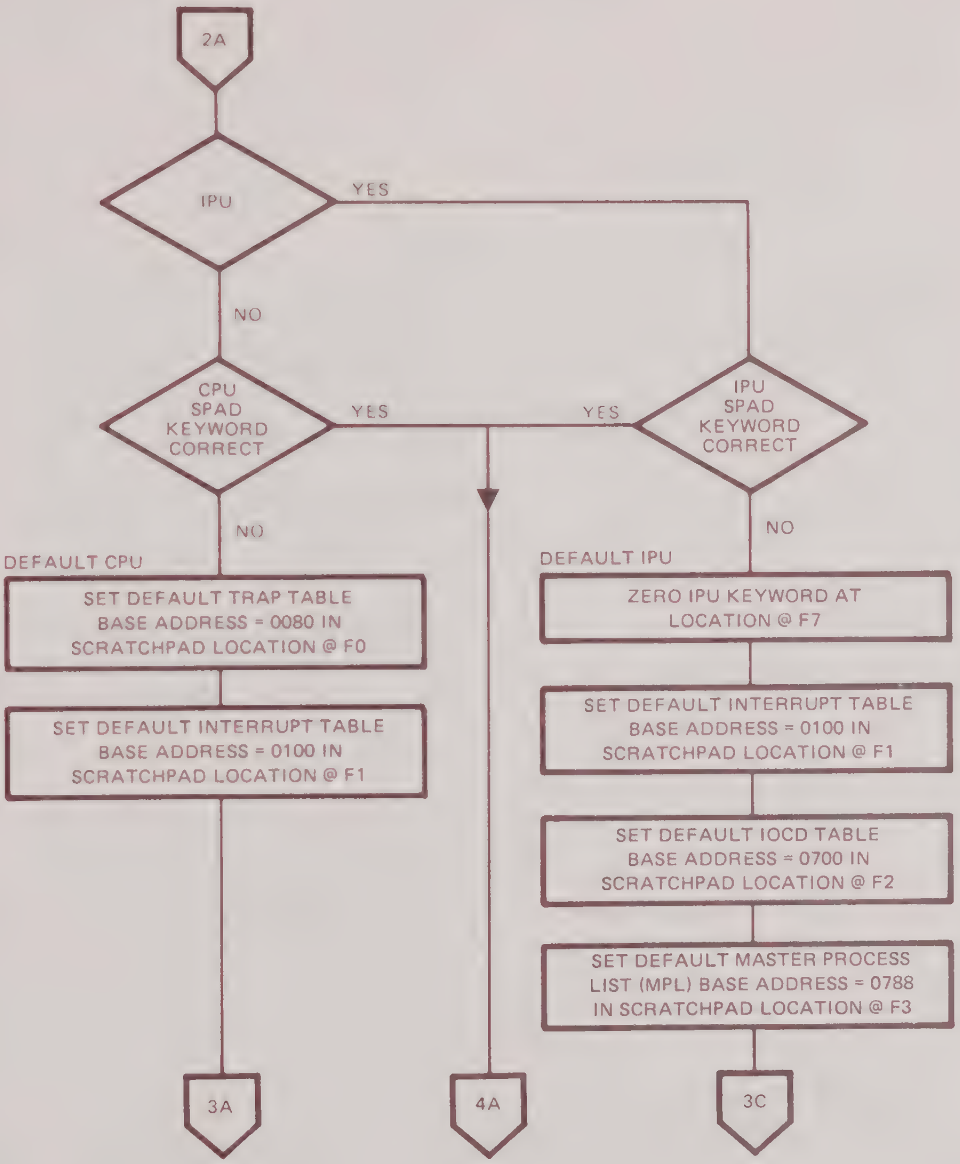

SYSTEM RESET
POWER UP
INITIAL PROGRAM LOAD
FIGURE 2-2
FIGURE 2-4
HARDWARE DEFINES PROCESSOR NO.1 AS CPU, IPU, OR OFF-LINE AND PROCESSOR NO. 2 AS IPU, CPU, OR OFF-LINE AS DETERMINED BY THE TURNKEY SWITCH
DEFAULT TO PROM MODE IF NOT ROMSIM MODE
CLEAR GPRS 0 THRU 7 IN CPU AND FPA
CLEAR MP2901, IE UNIT, AND FPA REGISTERS 8 THRU F
CLEAR CACHE
IPU
NO
YES
GENERATE SIGNAL IPU
GENERATE SIGNAL CPU
CLEAR BITS 0 THRU 3 AND BIT 16 IN DEVICE AND INTERRUPT ENTRIES (LOCATIONS @ 00 THRU @ EF)
ZERO SCRATCHPAD (LOCATIONS @ 100 THRU @ 10F
2A
830402

Figure 2-3. System Reset Flowchart (Sheet 1 of 11)


2A
IPU
YES
NO
CPU SPAD KEYWORD CORRECT
YES
YES
IPU SPAD KEYWORD CORRECT
NO
NO
DEFAULT CPU
SET DEFAULT TRAP TABLE BASE ADDRESS = 0080 IN SCRATCHPAD LOCATION @ F0
SET DEFAULT INTERRUPT TABLE BASE ADDRESS = 0100 IN SCRATCHPAD LOCATION @ F1
DEFAULT IPU
ZERO IPU KEYWORD AT LOCATION @ F7
SET DEFAULT INTERRUPT TABLE BASE ADDRESS = 0100 IN SCRATCHPAD LOCATION @ F1
SET DEFAULT IOCD TABLE BASE ADDRESS = 0700 IN SCRATCHPAD LOCATION @ F2
SET DEFAULT MASTER PROCESS LIST (MPL) BASE ADDRESS = 0788 IN SCRATCHPAD LOCATION @ F3
3A
4A
3C

NOTES:

THE SCRATCHPAD KEYWORD IS ONLY LOADED INTO THE SCRATCHPAD BY POWERUP AUTO RESTART OR A TRSC INSTRUCTION DURING SOFTWARE INITIALIZATION OF THE CPU SCRATCHPAD.

THE PRESENCE OF THE KEYWORD INDICATES THAT THE BASE ADDRESSES HAVE BEEN SET BY SOFTWARE AND MUST NOT BE DEFAULTED DURING SYSTEM RESET.

830411

Figure 2-3. System Reset Flowchart (Sheet 2 of 11)

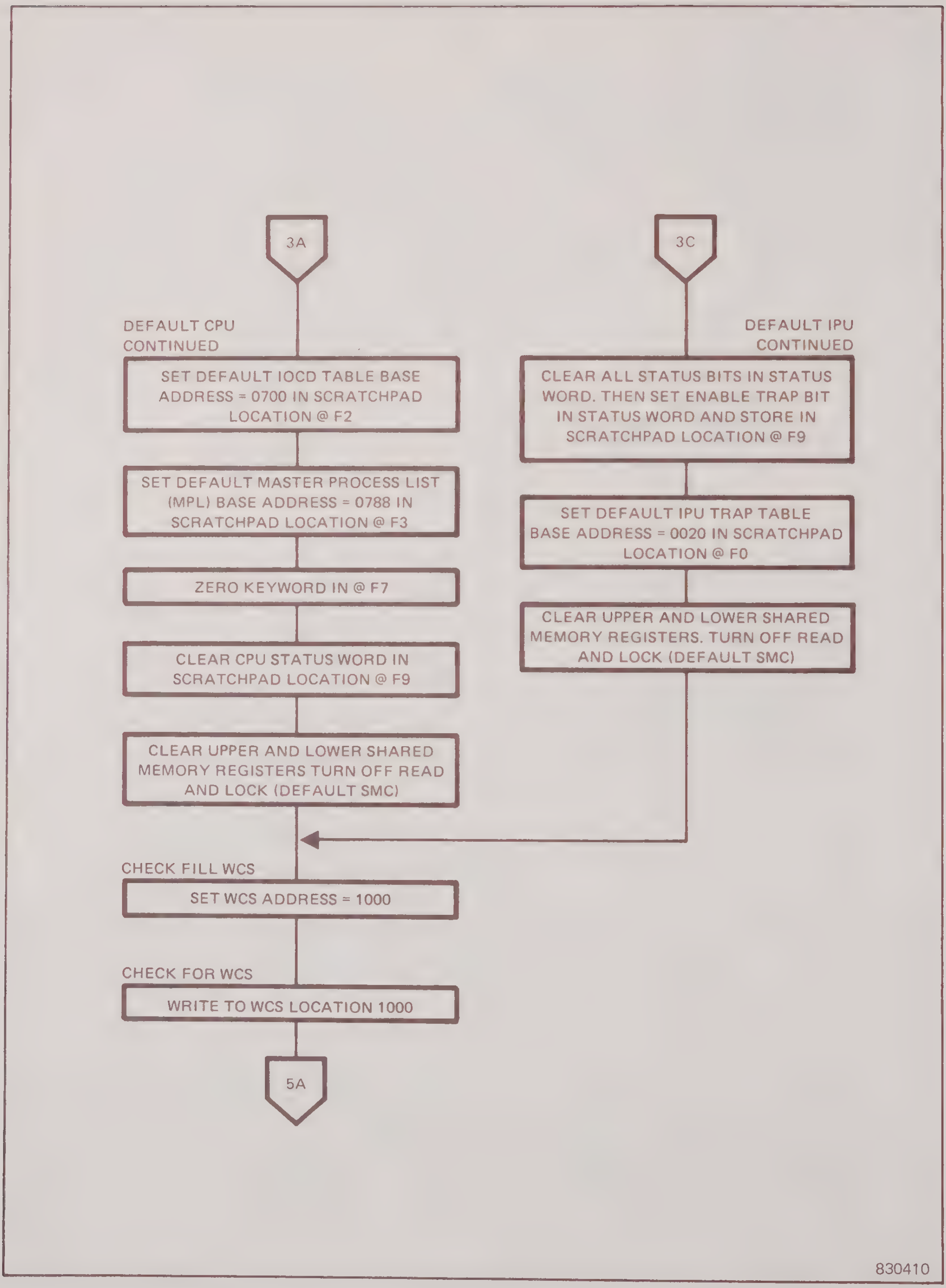

3A
3C
DEFAULT CPU CONTINUED
SET DEFAULT IOCD TABLE BASE ADDRESS = 0700 IN SCRATCHPAD LOCATION @ F2
SET DEFAULT MASTER PROCESS LIST (MPL) BASE ADDRESS = 0788 IN SCRATCHPAD LOCATION @ F3
ZERO KEYWORD IN @ F7
CLEAR CPU STATUS WORD IN SCRATCHPAD LOCATION @ F9
CLEAR UPPER AND LOWER SHARED MEMORY REGISTERS TURN OFF READ AND LOCK (DEFAULT SMC)
DEFAULT IPU CONTINUED
CLEAR ALL STATUS BITS IN STATUS WORD. THEN SET ENABLE TRAP BIT IN STATUS WORD AND STORE IN SCRATCHPAD LOCATION @ F9
SET DEFAULT IPU TRAP TABLE BASE ADDRESS = 0020 IN SCRATCHPAD LOCATION @ F0
CLEAR UPPER AND LOWER SHARED MEMORY REGISTERS. TURN OFF READ AND LOCK (DEFAULT SMC)
CHECK FILL WCS
SET WCS ADDRESS = 1000
CHECK FOR WCS
WRITE TO WCS LOCATION 1000
5A
830410

Figure 2-3. System Reset Flowchart (Sheet 3 of 11)

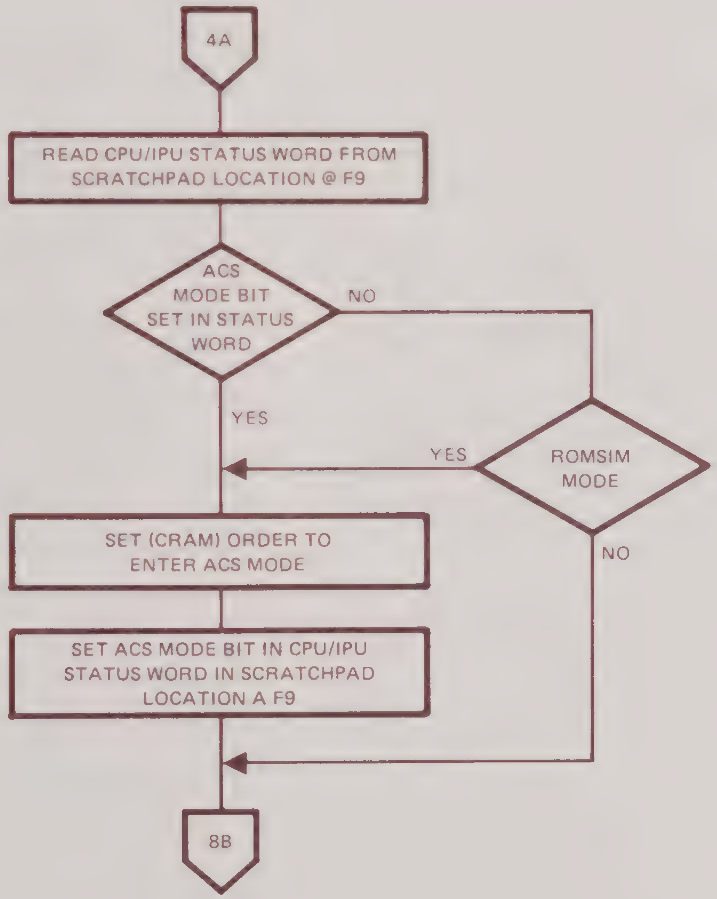

4A
READ CPU/IPU STATUS WORD FROM SCRATCHPAD LOCATION @ F9
ACS MODE BIT SET IN STATUS WORD
NO
YES
ROMSIM MODE
YES
NO
SET (CRAM) ORDER TO ENTER ACS MODE
SET ACS MODE BIT IN CPU/IPU STATUS WORD IN SCRATCHPAD LOCATION A F9
8B
830409

Figure 2-3. System Reset Flowchart (Sheet 4 of 11)

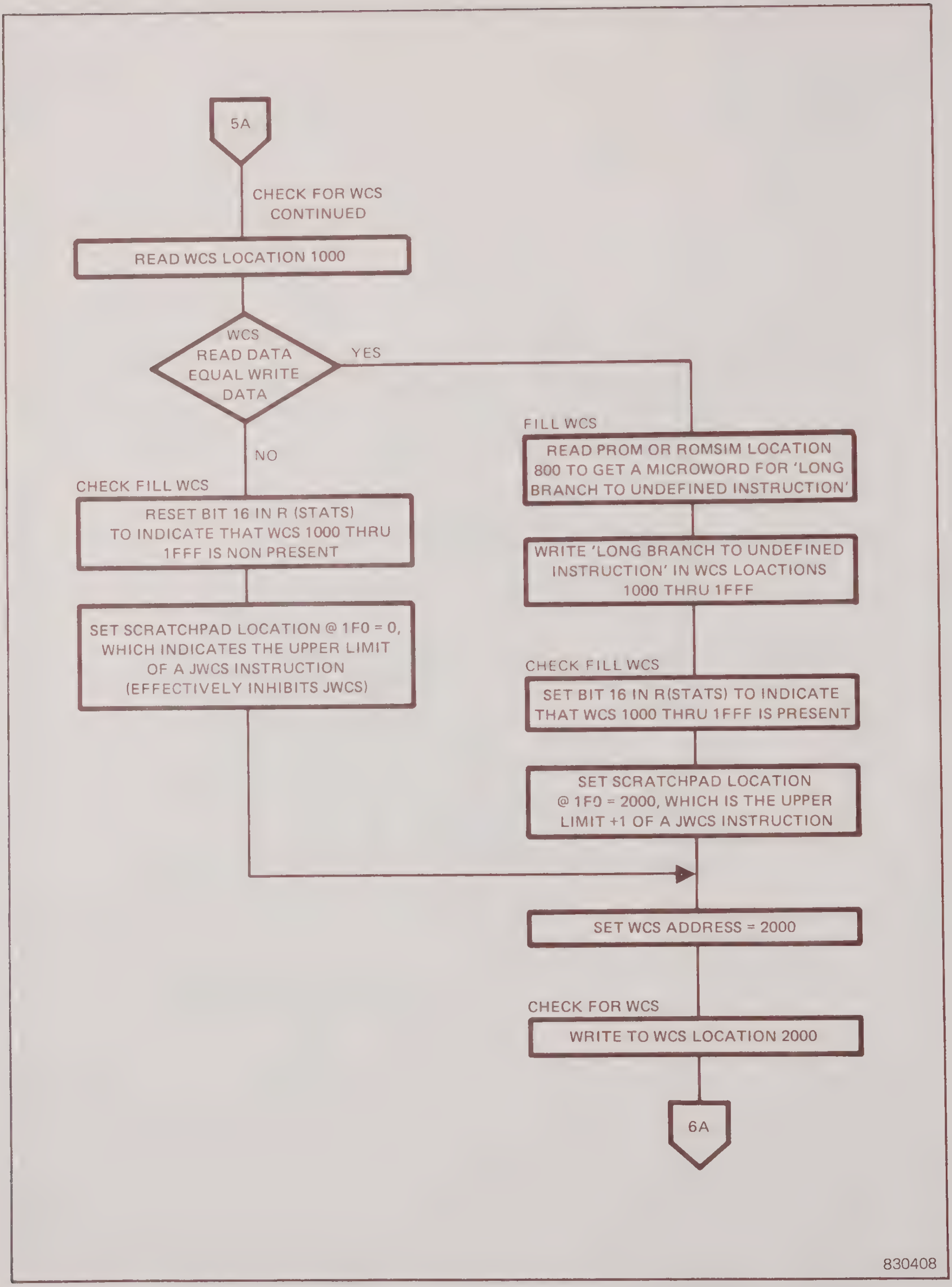

5A
CHECK FOR WCS CONTINUED
READ WCS LOCATION 1000
WCS READ DATA EQUAL WRITE DATA
YES
NO
FILL WCS
READ PROM OR ROMSIM LOCATION 800 TO GET A MICROWORD FOR 'LONG BRANCH TO UNDEFINED INSTRUCTION'
WRITE 'LONG BRANCH TO UNDEFINED INSTRUCTION' IN WCS LOACTIONS 1000 THRU 1FFF
CHECK FILL WCS
SET BIT 16 IN R(STATS) TO INDICATE THAT WCS 1000 THRU 1FFF IS PRESENT
SET SCRATCHPAD LOCATION @ 1F0 = 2000, WHICH IS THE UPPER LIMIT +1 OF A JWCS INSTRUCTION
CHECK FILL WCS
RESET BIT 16 IN R (STATS) TO INDICATE THAT WCS 1000 THRU 1FFF IS NON PRESENT
SET SCRATCHPAD LOCATION @ 1F0 = 0, WHICH INDICATES THE UPPER LIMIT OF A JWCS INSTRUCTION (EFFECTIVELY INHIBITS JWCS)
SET WCS ADDRESS = 2000
CHECK FOR WCS
WRITE TO WCS LOCATION 2000
6A
830408

Figure 2-3. System Reset Flowchart (Sheet 5 of 11)

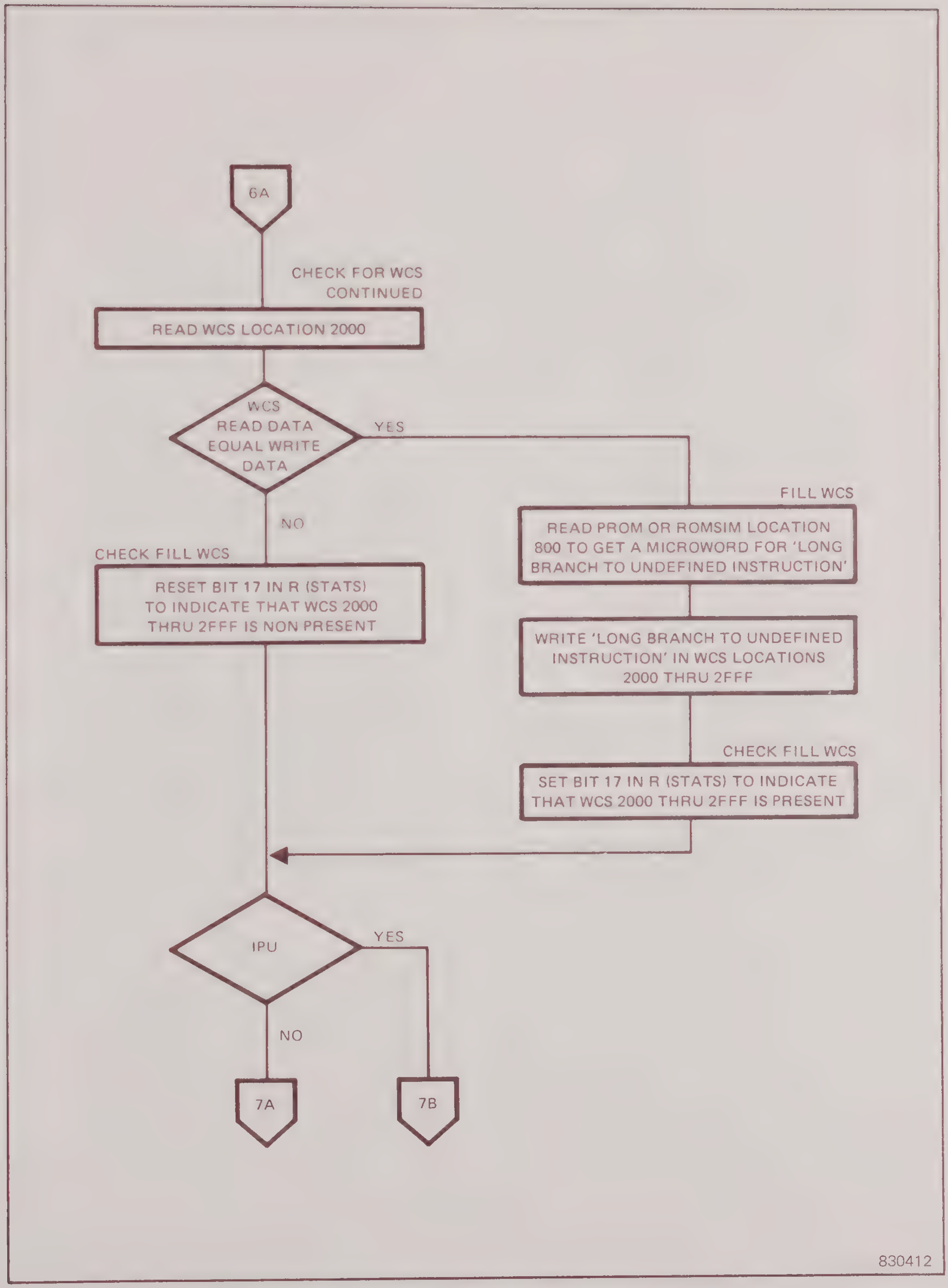

6A
CHECK FOR WCS CONTINUED
READ WCS LOCATION 2000
WCS READ DATA EQUAL WRITE DATA
YES
NO
FILL WCS
READ PROM OR ROMSIM LOCATION 800 TO GET A MICROWORD FOR 'LONG BRANCH TO UNDEFINED INSTRUCTION'
WRITE 'LONG BRANCH TO UNDEFINED INSTRUCTION' IN WCS LOCATIONS 2000 THRU 2FFF
CHECK FILL WCS
SET BIT 17 IN R (STATS) TO INDICATE THAT WCS 2000 THRU 2FFF IS PRESENT
CHECK FILL WCS
RESET BIT 17 IN R (STATS) TO INDICATE THAT WCS 2000 THRU 2FFF IS NON PRESENT
IPU
YES
NO
7A
7B
830412

Figure 2-3. System Reset Flowchart (Sheet 6 of 11)

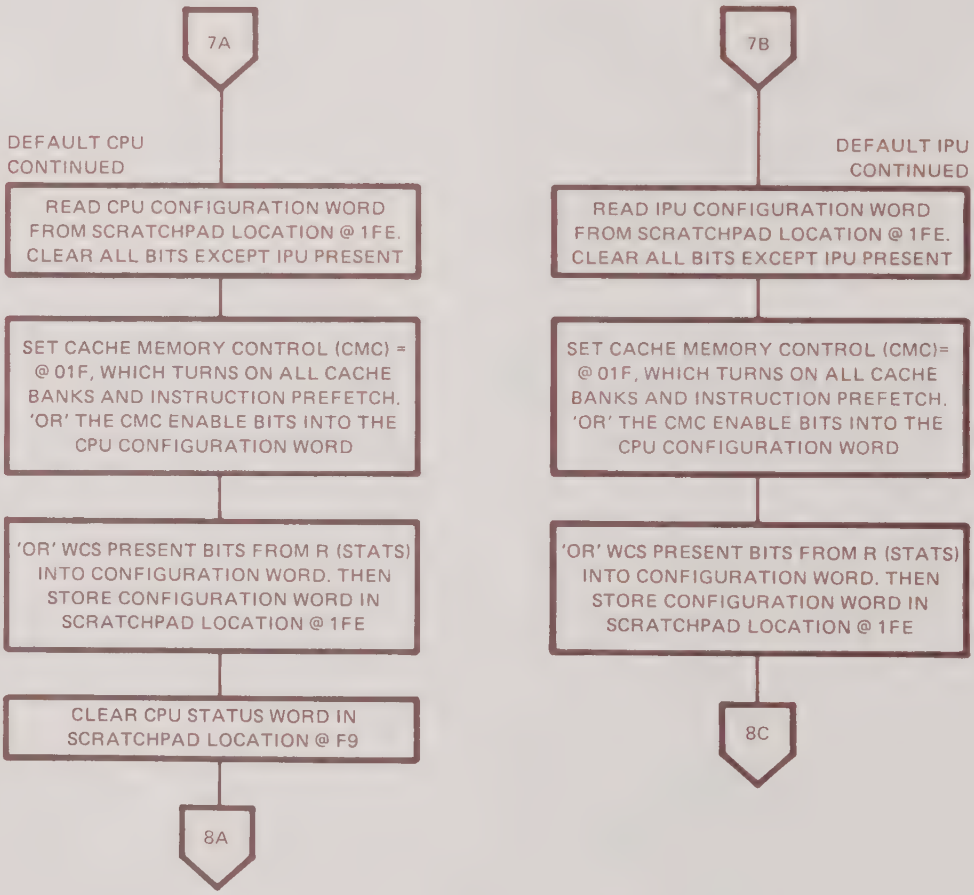

7A
DEFAULT CPU CONTINUED
READ CPU CONFIGURATION WORD FROM SCRATCHPAD LOCATION @ 1FE. CLEAR ALL BITS EXCEPT IPU PRESENT
SET CACHE MEMORY CONTROL (CMC) = @ 01F, WHICH TURNS ON ALL CACHE BANKS AND INSTRUCTION PREFETCH. 'OR' THE CMC ENABLE BITS INTO THE CPU CONFIGURATION WORD
'OR' WCS PRESENT BITS FROM R (STATS) INTO CONFIGURATION WORD. THEN STORE CONFIGURATION WORD IN SCRATCHPAD LOCATION @ 1FE
CLEAR CPU STATUS WORD IN SCRATCHPAD LOCATION @ F9
8A
7B
DEFAULT IPU CONTINUED
READ IPU CONFIGURATION WORD FROM SCRATCHPAD LOCATION @ 1FE. CLEAR ALL BITS EXCEPT IPU PRESENT
SET CACHE MEMORY CONTROL (CMC)= @ 01F, WHICH TURNS ON ALL CACHE BANKS AND INSTRUCTION PREFETCH. 'OR' THE CMC ENABLE BITS INTO THE CPU CONFIGURATION WORD
'OR' WCS PRESENT BITS FROM R (STATS) INTO CONFIGURATION WORD. THEN STORE CONFIGURATION WORD IN SCRATCHPAD LOCATION @ 1FE
8C
830413

Figure 2-3. System Reset Flowchart (Sheet 7 of 11)

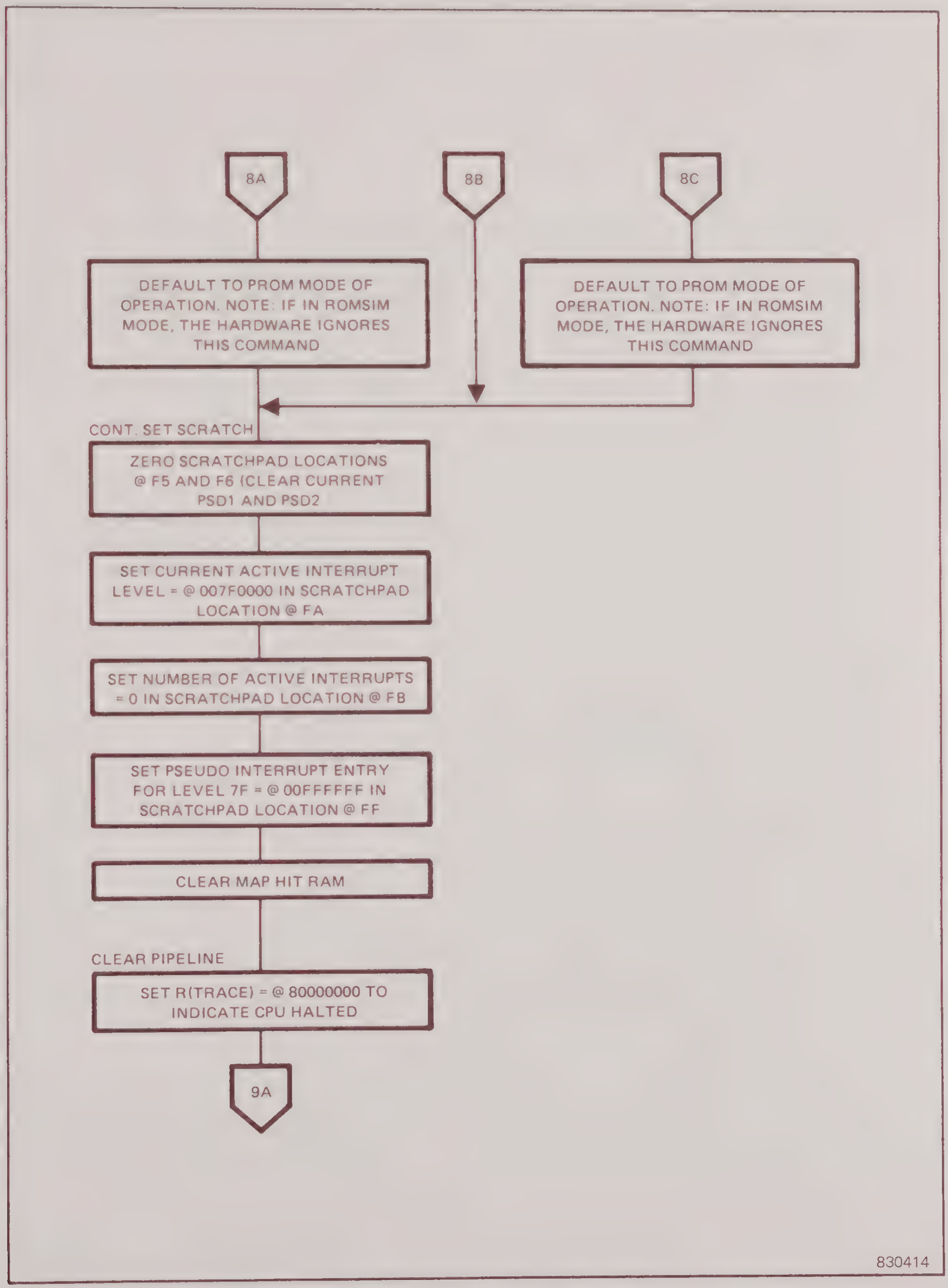

8A
8B
8C
DEFAULT TO PROM MODE OF OPERATION. NOTE: IF IN ROMSIM MODE, THE HARDWARE IGNORES THIS COMMAND
DEFAULT TO PROM MODE OF OPERATION. NOTE: IF IN ROMSIM MODE, THE HARDWARE IGNORES THIS COMMAND
CONT. SET SCRATCH
ZERO SCRATCHPAD LOCATIONS @ F5 AND F6 (CLEAR CURRENT PSD1 AND PSD2
SET CURRENT ACTIVE INTERRUPT LEVEL = @ 007F0000 IN SCRATCHPAD LOCATION @ FA
SET NUMBER OF ACTIVE INTERRUPTS = 0 IN SCRATCHPAD LOCATION @ FB
SET PSEUDO INTERRUPT ENTRY FOR LEVEL 7F = @ 00FFFFFF IN SCRATCHPAD LOCATION @ FF
CLEAR MAP HIT RAM
CLEAR PIPELINE
SET R(TRACE) = @ 80000000 TO INDICATE CPU HALTED
9A
830414

Figure 2-3. System Reset Flowchart (Sheet 8 of 11)

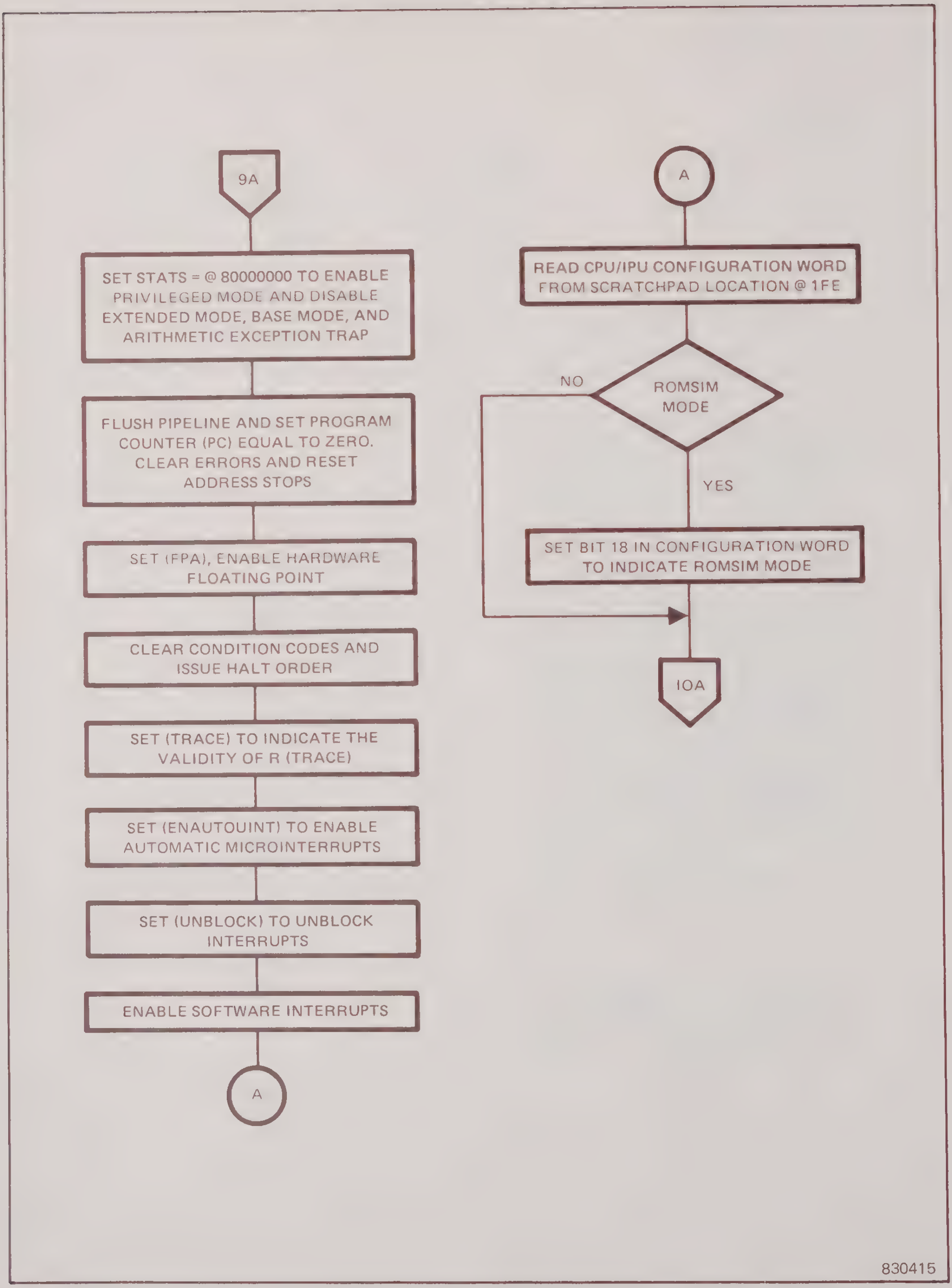
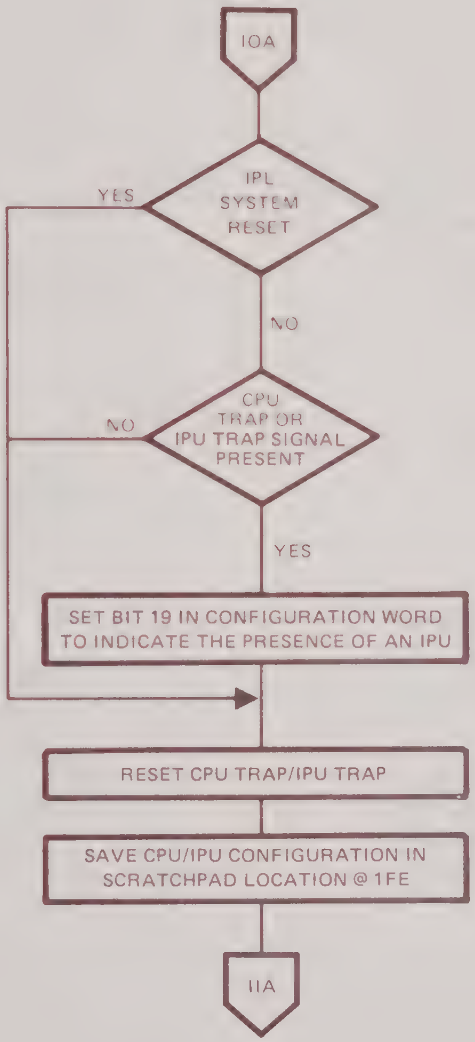

9A
SET STATS = @ 80000000 TO ENABLE PRIVILEGED MODE AND DISABLE EXTENDED MODE, BASE MODE, AND ARITHMETIC EXCEPTION TRAP
FLUSH PIPELINE AND SET PROGRAM COUNTER (PC) EQUAL TO ZERO. CLEAR ERRORS AND RESET ADDRESS STOPS
SET (FPA), ENABLE HARDWARE FLOATING POINT
CLEAR CONDITION CODES AND ISSUE HALT ORDER
SET (TRACE) TO INDICATE THE VALIDITY OF R (TRACE)
SET (ENAUTOUINT) TO ENABLE AUTOMATIC MICROINTERRUPTS
SET (UNBLOCK) TO UNBLOCK INTERRUPTS
ENABLE SOFTWARE INTERRUPTS
A
A
READ CPU/IPU CONFIGURATION WORD FROM SCRATCHPAD LOCATION @ 1FE
ROMSIM MODE
NO
YES
SET BIT 18 IN CONFIGURATION WORD TO INDICATE ROMSIM MODE
IOA
830415

Figure 2-3. System Reset Flowchart (Sheet 9 of 11)


IOA
IPL SYSTEM RESET
YES
NO
CPU TRAP OR IPU TRAP SIGNAL PRESENT
NO
YES
SET BIT 19 IN CONFIGURATION WORD TO INDICATE THE PRESENCE OF AN IPU
RESET CPU TRAP/IPU TRAP
SAVE CPU/IPU CONFIGURATION IN SCRATCHPAD LOCATION @ 1FE
IIA

830416

Figure 2-3. System Reset Flowchart (Sheet 10 of 11)

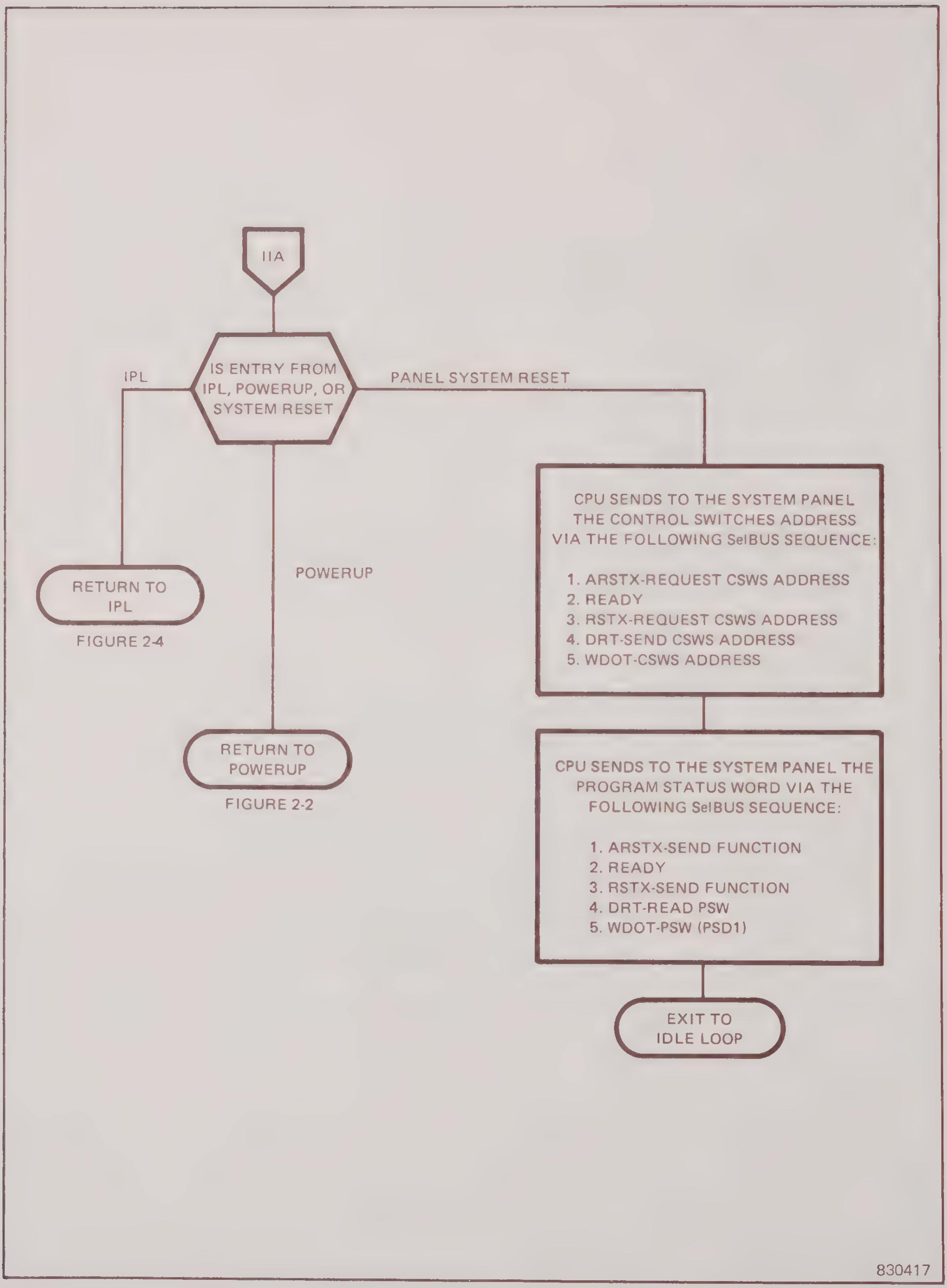

IIA
IS ENTRY FROM IPL, POWERUP, OR SYSTEM RESET
IPL
PANEL SYSTEM RESET
POWERUP
RETURN TO IPL
FIGURE 2-4
RETURN TO POWERUP
FIGURE 2-2
CPU SENDS TO THE SYSTEM PANEL THE CONTROL SWITCHES ADDRESS VIA THE FOLLOWING SelBUS SEQUENCE:
1. ARSTX-REQUEST CSWS ADDRESS
2. READY
3. RSTX-REQUEST CSWS ADDRESS
4. DRT-SEND CSWS ADDRESS
5. WDOT-CSWS ADDRESS
CPU SENDS TO THE SYSTEM PANEL THE PROGRAM STATUS WORD VIA THE FOLLOWING SelBUS SEQUENCE:
1. ARSTX-SEND FUNCTION
2. READY
3. RSTX-SEND FUNCTION
4. DRT-READ PSW
5. WDOT-PSW (PSD1)
EXIT TO IDLE LOOP
830417

Figure 2-3. System Reset Flowchart (Sheet 11 of 11)

g. The WCS, SMC, and CMC configuration word bits.

h. Cache memory control.

i. The enabled/disabled state of the hardware FPA.

j. The content of the ACS.

k. If the CPU/IPU was operating in the ACS mode prior to the system reset, the ACS mode of operation is restored.

7. The scratchpad current PSD1 and PSD2 in locations F5 and F6 are set to zero.

8. The current active interrupt level is set to 007F0000 in scratchpad location FA.

9. The number of active interrupts are set to zero in scratchpad location FB.

10. The pseudo interrupt entry for level 7F is set to 00FFFFFF in scratchpad location FF.

11. The map hit RAM is cleared.

12. The instruction pipeline is cleared and all pending errors are purged.

13. The CPU/IPU PSD Word 1 is set to 8000 0000 to indicate the privileged state with a program counter value of zero (CPU halted).

14. Interrupts are enabled and unblocked.

15. The HALT indicator is turned on.

16. The CPU/IPU power-fail, panel attention traps, and micro-interrupts are enabled.

17. The CPU/IPU initializes the address of the console control switches (CSW) by using a SelBUS CPU to panel communication sequence.

18. The CPU/IPU sends to the console the current PSD1 display via a CPU or IPU to panel SelBUS communication sequence. At reset, the PSD1 is 8000 0000.

19. The CPU/IPU enters the firmware idle loop and is ready to begin software instruction execution when a halt-to-run (run command) is received from the console.

20. The panels display the CPU and IPU PSD1 and the contents of the memory location addressed by PSD1 in a program status word (PSW) display.

2.5.2 Initial Program Load

The IPL function can only be executed while the CPU is in the halt mode and the halt indicator is on. The IPL cannot be executed while the CPU is in the run mode. The restriction to IPL execution is enforced by both the CPU and the IOP. IPL cannot be executed in the IPU.

For descriptive purposes, the IPL function can be divided into seven major sequences as follows:

1. The CPU receives the IPL signal (command) while halted and executing the firmware idle loop. The IPL command is enabled via the console.

2. The CPU executes a limited firmware system reset as shown within the figure 2-3 flowchart.

3. The CPU clears scratchpad location @FC to indicate a manual IPL.

4. The CPU initiates a SelBUS communications sequence with the IOP to obtain the IPL I/O device address.

5. The CPU initiates a SelBUS communication sequence with the IPL I/O device's channel to determine the channel's SelBUS I/O protocol; either class E or class F.

6. The CPU initiates a SelBUS sequence for an IPL-start I/O. The IPL-start I/O specifies a binary data read, starting at memory location zero.

7. The CPU waits for the IPL I/O data transfer to be completed, indicating that the software bootstrap program has been loaded into memory from the IPL device.

8. The CPU fills the instruction pipeline and begins execution of the software bootstrap program. The software bootstrap must initialize the remaining elements of the CPU and system.

NOTE

Memory location 780 is not cleared during manual IPL.

Figure 2-4 is a flowchart of the IPL sequence. For auto-IPL, figures 2-2 and 2-3 each contain portions of the auto-IPL sequence.

2.5.2.1 Firmware Reset

During IPL, the CPU performs a limited firmware reset. Since a hardware reset is not performed, the I/O channels and devices are not reset. Therefore, the manual IPL must always be preceded by a system reset function to assure that all I/O channels are reset.

The limited firmware reset executes the functions of a normal firmware reset, except it does not clear the software general-purpose registers. Since the GPRs are not cleared, they may be loaded prior to an IPL by a panel function. If an IPL is executed, without an intervening system reset, the contents of the preloaded GPRs will be passed through the IPL sequence to the software bootstrap program at the conclusion of the IPL sequence.

The limited firmware reset causes all scratchpad base addresses to be set to their default values. This assures that an automatic trap halt will function correctly in an IPL and software bootstrap sequence.

All operating features of the 32/67 (WCS, SMC, ACS/PROM, and CMC) are set to their default values as described under the System Reset heading.

2.5.2.2 IPL Device Address

An IPL device address consists of a 15-bit right-justified number. The most-significant seven bits are the I/O channel physical SelBUS address, and the eight least-significant bits are the device address or subaddress.

The IPL device address may be acquired from one of the four sources which follow:

1. In auto IPL, the IPL address is sourced from 16 auto-IPL address jumpers located on the MS board.

2. A console IPL address zero may be entered via the console. IPL address zero is meaningless, but causes the CPU firmware to use the default IPL address located in scratchpad location F4. This default IPL address is the same address used for the last previously successful IPL since a power-up sequence had occurred. However, if this is the first IPL following a power up, the scratchpad default IPL address may be unreliable.

3. A 15-bit, nonzero, right-justified, IPL address may be entered with the IPL command from the console. This IPL address is transmitted to the CPU which does not check the address validity. If the IPL at this address is successful, the address is stored in scratchpad location F4 as the new default IPL address.

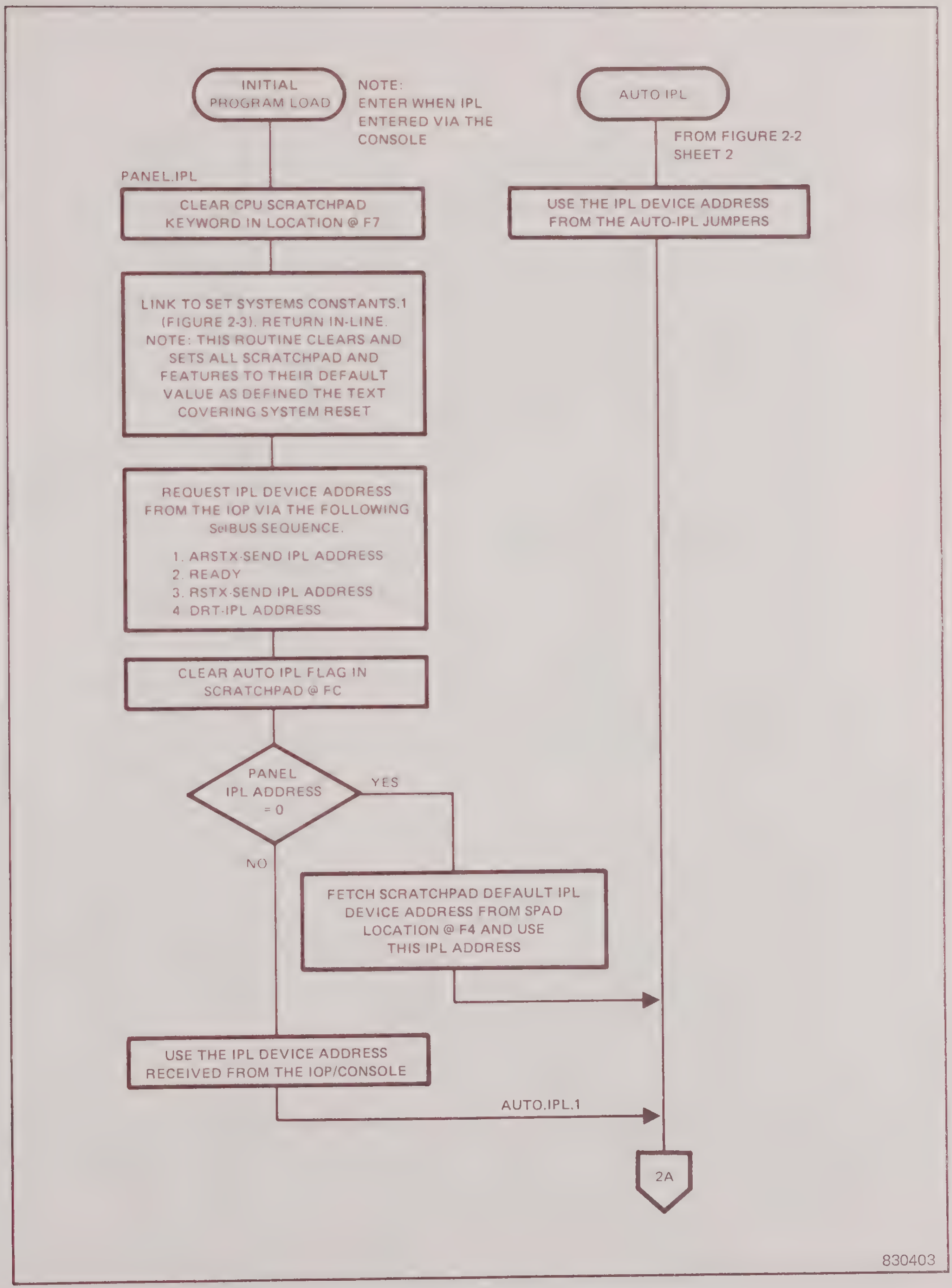

INITIAL PROGRAM LOAD
NOTE: ENTER WHEN IPL ENTERED VIA THE CONSOLE
AUTO IPL
FROM FIGURE 2-2 SHEET 2
PANEL.IPL
CLEAR CPU SCRATCHPAD KEYWORD IN LOCATION @ F7
USE THE IPL DEVICE ADDRESS FROM THE AUTO-IPL JUMPERS
LINK TO SET SYSTEMS CONSTANTS.1 (FIGURE 2-3). RETURN IN-LINE. NOTE: THIS ROUTINE CLEARS AND SETS ALL SCRATCHPAD AND FEATURES TO THEIR DEFAULT VALUE AS DEFINED THE TEXT COVERING SYSTEM RESET
REQUEST IPL DEVICE ADDRESS FROM THE IOP VIA THE FOLLOWING SelBUS SEQUENCE.
1. ARSTX-SEND IPL ADDRESS
2. READY
3. RSTX-SEND IPL ADDRESS
4 DRT-IPL ADDRESS
CLEAR AUTO IPL FLAG IN SCRATCHPAD @ FC
PANEL IPL ADDRESS = 0
YES
NO
FETCH SCRATCHPAD DEFAULT IPL DEVICE ADDRESS FROM SPAD LOCATION @ F4 AND USE THIS IPL ADDRESS
USE THE IPL DEVICE ADDRESS RECEIVED FROM THE IOP/CONSOLE
AUTO.IPL.1
2A
830403

Figure 2-4. Initial Program Load Flowchart (sheet 1 of 5)

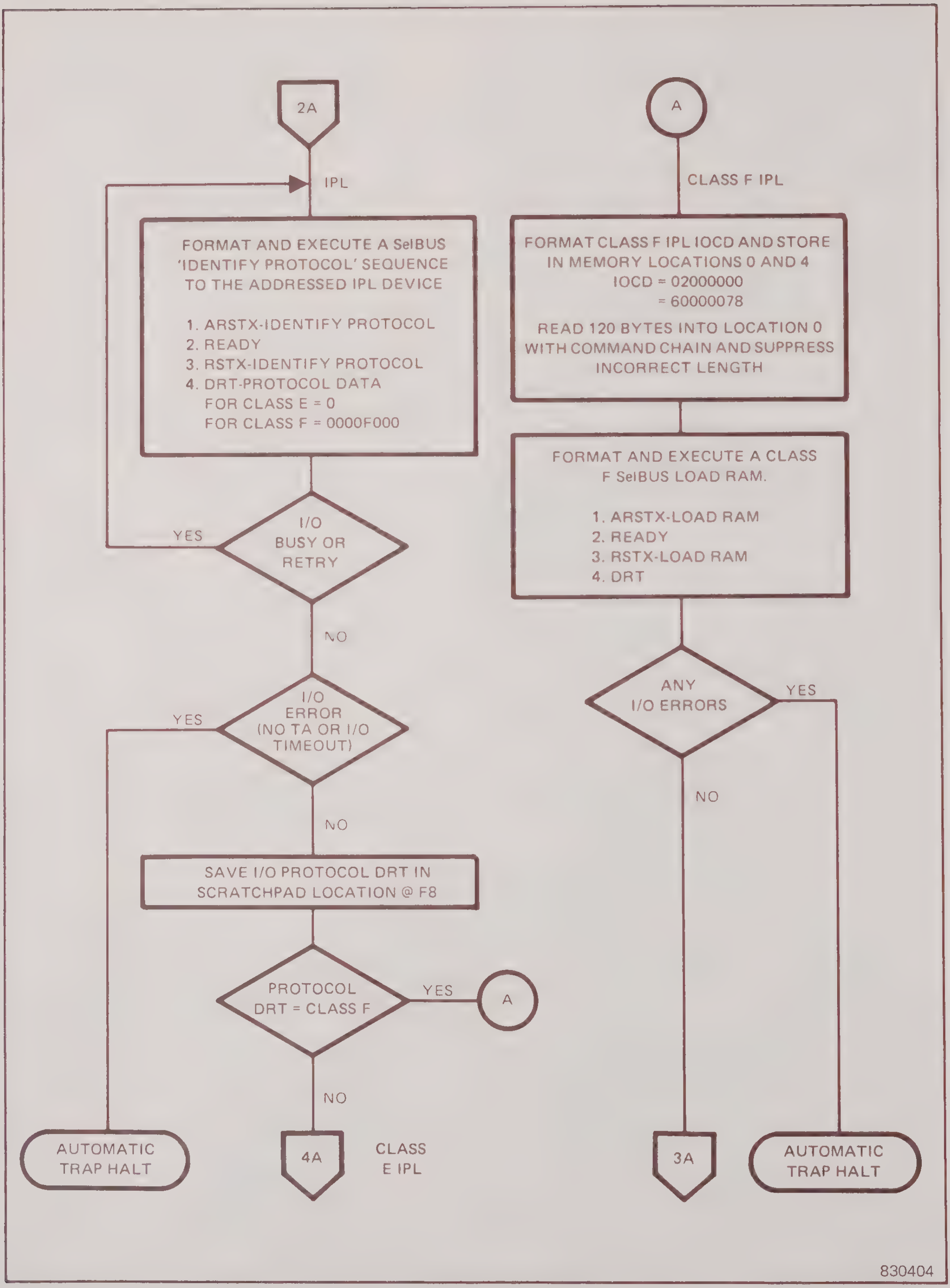

2A
IPL
FORMAT AND EXECUTE A SelBUS 'IDENTIFY PROTOCOL' SEQUENCE TO THE ADDRESSED IPL DEVICE
1. ARSTX-IDENTIFY PROTOCOL
2. READY
3. RSTX-IDENTIFY PROTOCOL
4. DRT-PROTOCOL DATA
FOR CLASS E = 0
FOR CLASS F = 0000F000
YES
I/O BUSY OR RETRY
NO
YES
I/O ERROR (NO TA OR I/O TIMEOUT)
NO
SAVE I/O PROTOCOL DRT IN SCRATCHPAD LOCATION @ F8
PROTOCOL DRT = CLASS F
YES
A
NO
AUTOMATIC TRAP HALT
4A
CLASS E IPL
A
CLASS F IPL
FORMAT CLASS F IPL IOCD AND STORE IN MEMORY LOCATIONS 0 AND 4
IOCD = 02000000
= 60000078
READ 120 BYTES INTO LOCATION 0 WITH COMMAND CHAIN AND SUPPRESS INCORRECT LENGTH
FORMAT AND EXECUTE A CLASS F SelBUS LOAD RAM.
1. ARSTX-LOAD RAM
2. READY
3. RSTX-LOAD RAM
4. DRT
ANY I/O ERRORS
YES
NO
3A
AUTOMATIC TRAP HALT
830404

Figure 2-4. Initial Program Load Flowchart (sheet 2 of 5)

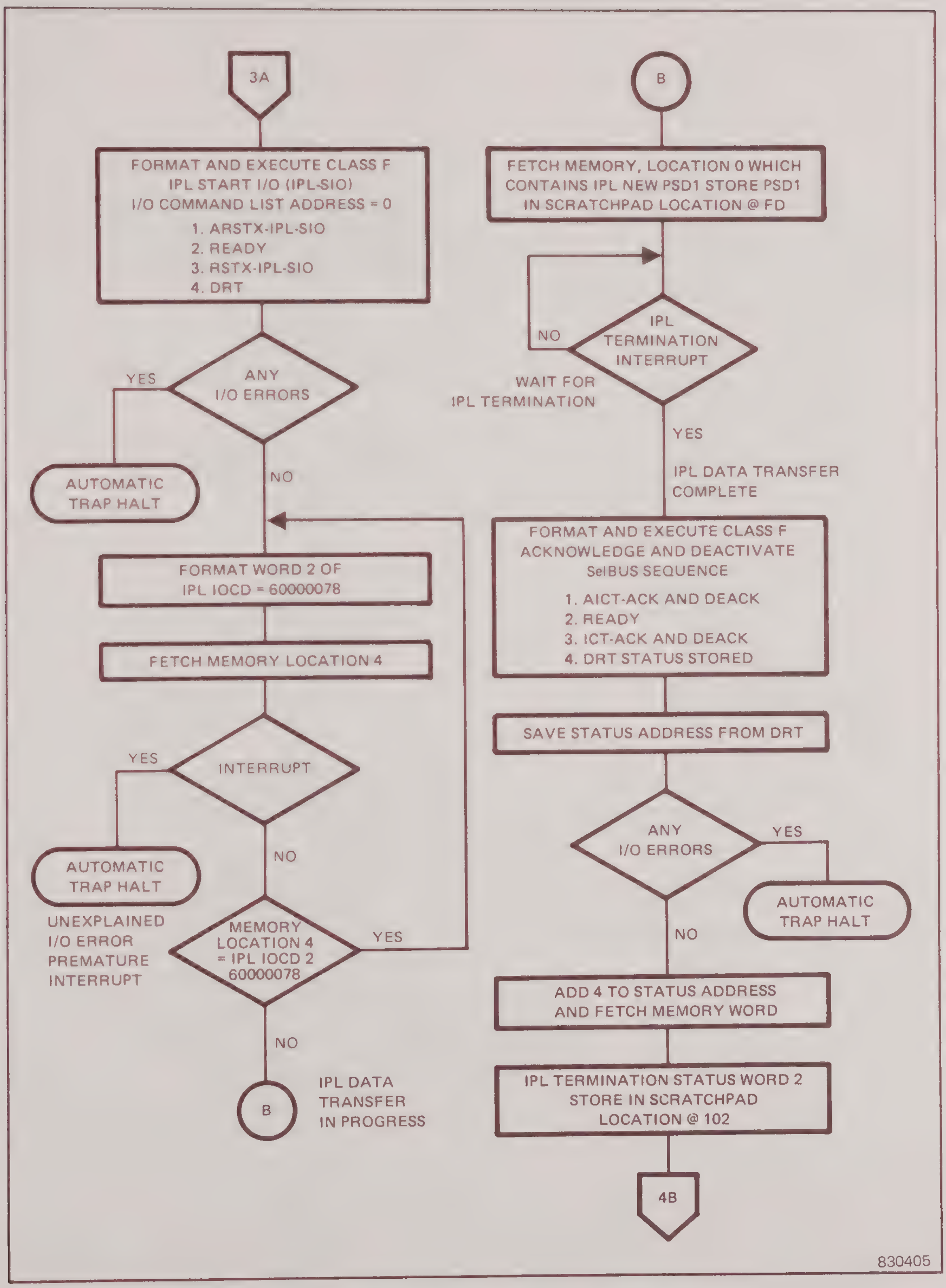
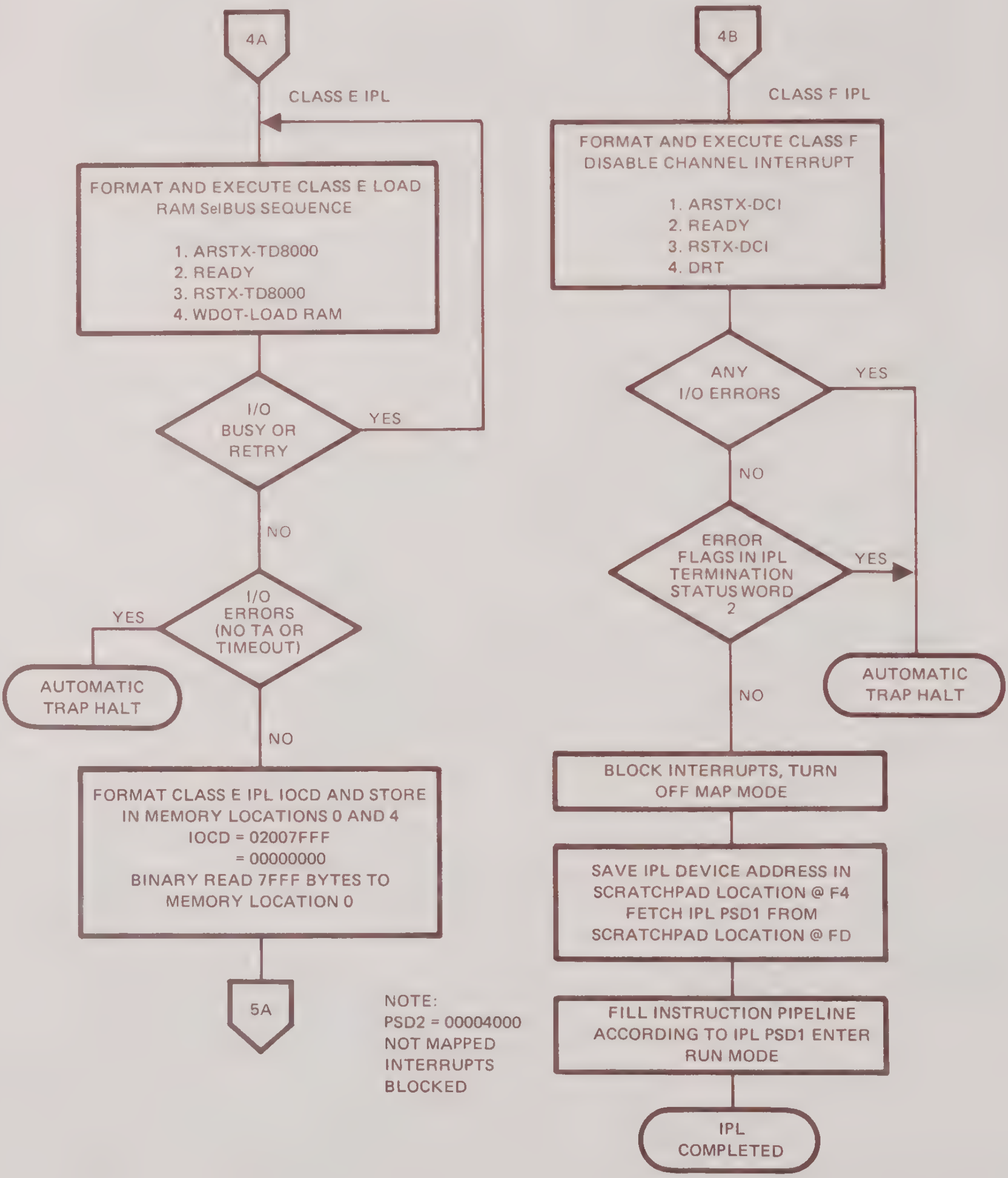
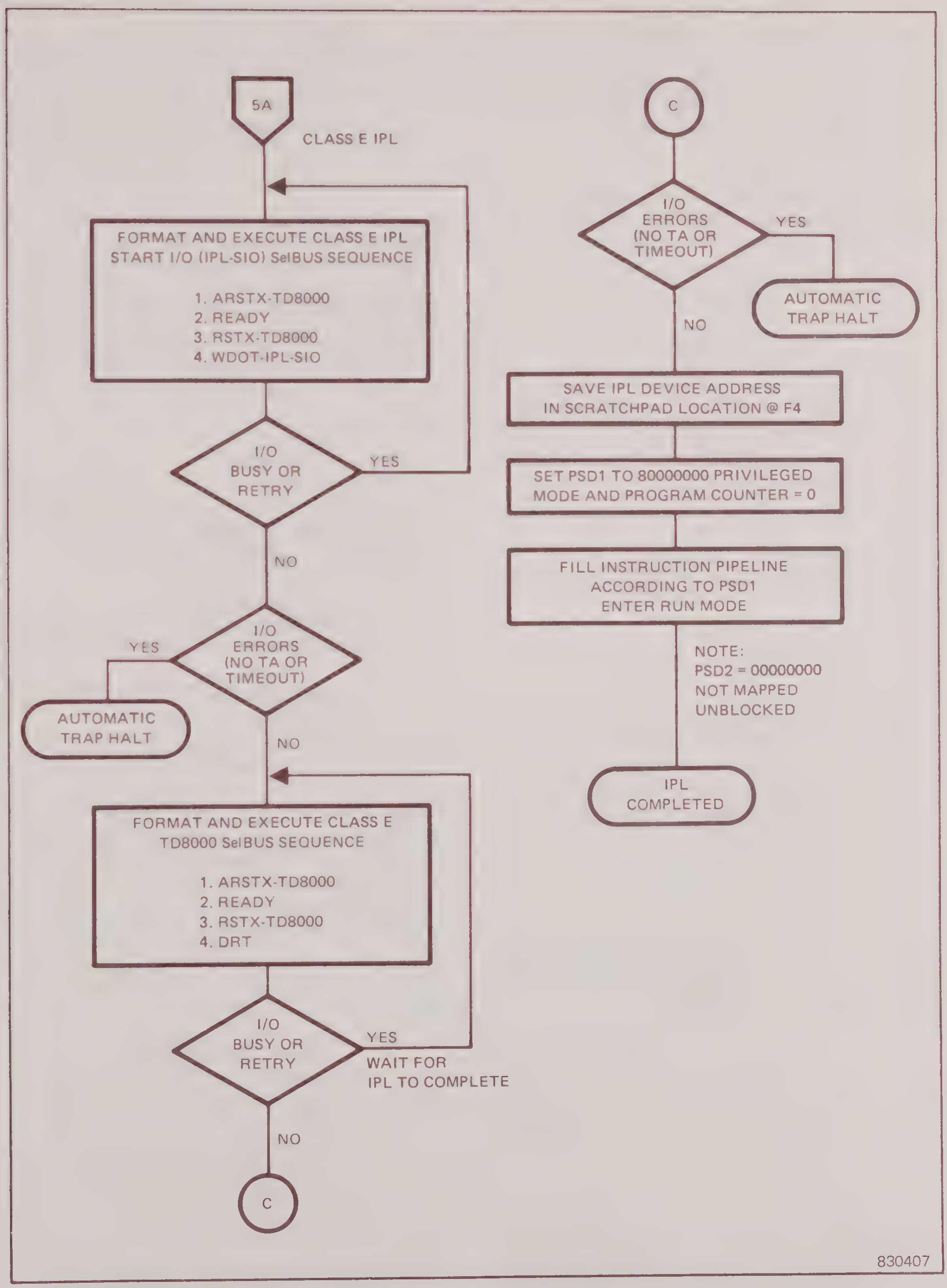

3A
FORMAT AND EXECUTE CLASS F IPL START I/O (IPL-SIO) I/O COMMAND LIST ADDRESS = 0
1. ARSTX-IPL-SIO
2. READY
3. RSTX-IPL-SIO
4. DRT
ANY I/O ERRORS
YES
AUTOMATIC TRAP HALT
NO
FORMAT WORD 2 OF IPL IOCD = 60000078
FETCH MEMORY LOCATION 4
INTERRUPT
YES
AUTOMATIC TRAP HALT
UNEXPLAINED I/O ERROR PREMATURE INTERRUPT
NO
MEMORY LOCATION 4 = IPL IOCD 2 60000078
YES
NO
B
IPL DATA TRANSFER IN PROGRESS
B
FETCH MEMORY, LOCATION 0 WHICH CONTAINS IPL NEW PSD1 STORE PSD1 IN SCRATCHPAD LOCATION @ FD
NO
IPL TERMINATION INTERRUPT
WAIT FOR IPL TERMINATION
YES
IPL DATA TRANSFER COMPLETE
FORMAT AND EXECUTE CLASS F ACKNOWLEDGE AND DEACTIVATE SelBUS SEQUENCE
1. AICT-ACK AND DEACK
2. READY
3. ICT-ACK AND DEACK
4. DRT STATUS STORED
SAVE STATUS ADDRESS FROM DRT
ANY I/O ERRORS
YES
AUTOMATIC TRAP HALT
NO
ADD 4 TO STATUS ADDRESS AND FETCH MEMORY WORD
IPL TERMINATION STATUS WORD 2 STORE IN SCRATCHPAD LOCATION @ 102
4B
830405

Figure 2-4. Initial Program Load Flowchart (sheet 3 of 5)


4A
CLASS E IPL
FORMAT AND EXECUTE CLASS E LOAD RAM SelBUS SEQUENCE
1. ARSTX-TD8000
2. READY
3. RSTX-TD8000
4. WDOT-LOAD RAM
I/O BUSY OR RETRY
YES
NO
I/O ERRORS (NO TA OR TIMEOUT)
YES
AUTOMATIC TRAP HALT
NO
FORMAT CLASS E IPL IOCD AND STORE IN MEMORY LOCATIONS 0 AND 4
IOCD = 02007FFF
= 00000000
BINARY READ 7FFF BYTES TO MEMORY LOCATION 0
5A
NOTE:
PSD2 = 00004000
NOT MAPPED
INTERRUPTS
BLOCKED
4B
CLASS F IPL
FORMAT AND EXECUTE CLASS F DISABLE CHANNEL INTERRUPT
1. ARSTX-DCI
2. READY
3. RSTX-DCI
4. DRT
ANY I/O ERRORS
YES
NO
ERROR FLAGS IN IPL TERMINATION STATUS WORD 2
YES
AUTOMATIC TRAP HALT
NO
BLOCK INTERRUPTS, TURN OFF MAP MODE
SAVE IPL DEVICE ADDRESS IN SCRATCHPAD LOCATION @ F4
FETCH IPL PSD1 FROM SCRATCHPAD LOCATION @ FD
FILL INSTRUCTION PIPELINE ACCORDING TO IPL PSD1 ENTER RUN MODE
IPL COMPLETED

830406

Figure 2-4. Initial Program Load Flowchart (sheet 4 of 5)



5A
CLASS E IPL
FORMAT AND EXECUTE CLASS E IPL START I/O (IPL-SIO) SelBUS SEQUENCE
1. ARSTX-TD8000
2. READY
3. RSTX-TD8000
4. WDOT-IPL-SIO
I/O BUSY OR RETRY
YES
NO
I/O ERRORS (NO TA OR TIMEOUT)
YES
AUTOMATIC TRAP HALT
NO
FORMAT AND EXECUTE CLASS E TD8000 SelBUS SEQUENCE
1. ARSTX-TD8000
2. READY
3. RSTX-TD8000
4. DRT
I/O BUSY OR RETRY
YES
WAIT FOR IPL TO COMPLETE
NO
C
C
I/O ERRORS (NO TA OR TIMEOUT)
YES
AUTOMATIC TRAP HALT
NO
SAVE IPL DEVICE ADDRESS IN SCRATCHPAD LOCATION @ F4
SET PSD1 TO 80000000 PRIVILEGED MODE AND PROGRAM COUNTER = 0
FILL INSTRUCTION PIPELINE ACCORDING TO PSD1 ENTER RUN MODE
NOTE:
PSD2 = 00000000
NOT MAPPED
UNBLOCKED
IPL COMPLETED
830407

Figure 2-4. Initial Program Load Flowchart (sheet 5 of 5)

4. In manual IPL sequences, if the IOP panel function is inoperable or nonpresent, the CPU will generate a physical and subaddress of 0100 hex, and then attempt to IPL from this address.

Any I/O device may be used for IPL, providing that the device, its channel, and controller meet the following criteria:

1. Binary I/O read is supported.
2. The channel supports the SelBUS identify I/O protocol sequence.
3. The channel and controller support the SelBUS IPL-start I/O sequence and IPL input/output command doubleword.
4. The I/O media can be automatically or manually positioned to the start of the IPL records.

2.5.2.3 IPL Identify I/O Protocol

The CPU initiates the SelBUS identify I/O protocol communications sequence in order to determine the I/O protocol of the addressed IPL channel, controller, and device as either class E or class F. The I/O protocol of the IPL channel determines the type and rules of further SelBUS communications during the SelBUS sequence.

In general, class F IPL is considered more accurate and adaptable than class E IPL for the following reasons:

1. Class F IPL rules are universally interpreted by the CPU and I/O channel; whereas, class E rules are interpreted differently by different I/O channels.
2. Class F rules provide for command and data chaining which allows IPL to be adapted to variable or fixed length record devices.
3. Class F rules provide for a termination status transfer to the CPU which informs the CPU of the integrity of the data loaded into memory. The CPU aborts the IPL sequence if any IPL I/O errors are present.

The IPL identify protocol data return transfer (DRT) is stored into scratchpad location F8. Bits 16 through 19, equal to F, indicate a class F protocol. Any other configuration in bits 16 through 19 is class E protocol. Bits 0 through 15 and 20 through 31 are I/O channel dependent bits.

2.5.2.4 IPL IOCDs

To control the IPL data transfer, the CPU formats and stores into memory locations 0 and 4 an input/output command doubleword. In class F rules, the IOCD specifies the following:

1. A binary read command
2. A data transfer starting address of 0
3. A data transfer count of 120 bytes
4. A flag byte that specifies command chaining and suppress incorrect length indicator.

Subsequent IOCDs to complete the command chain and an input/output command list (IOCL) must be read into memory from the IPL device media. Table 2-3 illustrates the contents of memory during the class F IPL sequence.

In class E rules the IOCD specifies the following:

1. A binary read command
2. A data transfer starting address of 0
3. A data transfer count of 7FFF transfers. (The interpretation of the transfer count as byte, halfword, or word is channel dependent.)

Table 2-4 illustrates the contents of memory during class E IPL sequence.

Table 2-3
Class F IPL Memory Contents

Memory contents before IPL-start I/O			
Location	**Contents**		
0	0200 0000	CPU-formatted IOCD	
4	6000 0078		
8	Irrelevant		
Memory contents after IPL data transfer but before IPL termination (all memory data has been read from the IPL device)			
Location	**Contents**		
0	8000 00XX PSD word 1	PSD	
4	0000 4000 PSD word 2		
8	IOCD word 1	IOCD	IOCL
C	IOCD word 2		
XX-8	Termination IOCD word 1		
XX-4	Termination IOCD word 2		
XX	Instruction	Software bootstrap	
XX+4	Instruction		
Memory contents after IPL termination			
Location	**Contents**		
0	YY00 00XX Status word 1	Termination status doubleword stored by the IPL channel. YY is the subaddress of the IPL device and XX is the next IOCD address	
4	000C 0000 Status word 2		
8	IOCD word 1	IOCD	IOCL
C	IOCD word 2		
XX-8	Termination IOCD word 1	IOCD	
XX-4	Termination IOCD word 2		
XX	Instruction	Software bootstrap	
XX+4	Instruction		

Note: Some F class channels post IPL termination status in memory at locations other than location zero. In these cases, the IPL PSD is not overwritten with termination status.

Table 2-4
Class E IPL Memory Contents

<table>
<tr><td colspan="3">Memory contents before IPL-start I/O</td></tr>
<tr><td>Location</td><td colspan="2">Contents</td></tr>
<tr><td>0</td><td>0200 7FFF</td><td>CPU-formatted IOCD</td></tr>
<tr><td>4</td><td>000 0000</td><td></td></tr>
<tr><td>8</td><td>Irrelevant</td><td></td></tr>
<tr><td colspan="3">Memory contents after IPL data transfer (all memory data has been read from the IPL device</td></tr>
<tr><td>Location</td><td colspan="2">Contents</td></tr>
<tr><td>0</td><td>Instruction</td><td></td></tr>
<tr><td>4</td><td>Instruction</td><td></td></tr>
<tr><td></td><td></td><td>Software Bootstrap</td></tr>
</table>

Table 2-5 illustrates the format of the first IPL record in class F IPL.

Table 2-5
Class F First IPL Record Format

<table>
<tr><th>Byte</th><th>Contents</th></tr>
<tr><td>0 through 3</td><td>CPU program status doubleword (PSD) word 1</td></tr>
<tr><td>4 through 7</td><td>CPU PSD word 2</td></tr>
<tr><td>8 through 15</td><td>Input/output command doubleword
NOTES
1. This IOCD may complete the chain initiated by the CPU IPL IOCD or may specify additional chaining.
2. If additional chaining is specified, additional IOCDs must be supplied starting in byte 16.</td></tr>
<tr><td>16 through (XX-1)</td><td>Additional IOCDs as required</td></tr>
<tr><td>XX through End</td><td>Software bootstrap</td></tr>
<tr><td colspan="2">NOTE
The bootstrap can be in a subsequent record, read by the IPL I/O command list.</td></tr>
</table>

2.5.2.5 IPL Start I/O

The IPL start I/O (IPL-SIO) is a CPU-initiated, SelBUS communication sequence to the IPL channel and device. The start I/O initiates the IPL data transfer and specifies the memory locations of the IOCD. Some I/O channels actually perform implicit media positioning as a result of the IPL start I/O sequence. For example, the disc channel executes a seek track zero, head zero, and sector zero as a result of the IPL start I/O sequence.

The IPL start I/O sequence is always preceded by a CPU initiated SelBUS load RAM sequence which informs the I/O channel of its SelBUS identity with respect to SelBUS physical (channel) address and interrupt level.

For IPL sequence, a pseudo interrupt level of 7F is assigned to the I/O channel; however, this interrupt level is only used to terminate class F IPL sequences and is not used at all in class E sequences.

2.5.2.6 IPL Data Transfers

The IPL data transfer is controlled by the IOCD stored at memory location zero. The IOCD specifies the command (binary read), the starting memory data transfer address (location 0), and the transfer count. In class F IOCDs, the flag byte specifies command chaining and suppress incorrect length indicator.

In class E IPL sequences, the amount of data transferred is dependent upon the I/O channel interpretation of the IOCD transfer count. Some class E channels terminate data transfer on transfer count zero or end of record, whichever comes first. Other class E channels will read multiple records until the IPL transfer count of 7FFF is exhausted. There is no problem for variable length record media, such as magnetic tape, since the IPL record can be up to 7FFF bytes in length. However, a problem occurs for fixed length record media, such as cards or disc, since the bootstrap program cannot determine the amount of data read into memory. Therefore, generalized IPL software bootstrap programs do not exist for class E IPL, and each bootstrap program is custom-tailored to the IPL channel, controller, device, and media.

In class F IPL sequences, the CPU formats and stores the first IOCD into memory locations 0 and 4 to form an input/output command list (IOCL). The remaining IOCDs to complete the IOCL are read into memory from the IPL media. The first IOCD controls the data transfer of the first 120 bytes of data. A minimum of one additional IOCD, required to complete the command chain started by the first IOCD, must be read into memory starting at location 8. The first 8 bytes of the IPL record must contain a CPU program status doubleword (PSD) that indicates the memory starting location of the software bootstrap program. The CPU PSD will overlay the first IOCD in memory when the IPL record is transferred to memory.

Table 2-5 illustrates the format of the first IPL record in class F IPL.

In general, the class F IOCL may consist of any IOCDs that are required to read the entire bootstrap program into memory. The IOCD commands may include the channel initialization (INCH) command and media positioning commands. These commands do not cause attention status indicators which cause the premature termination of the IOCL chaining function. If the IOCL contains an INCH command, the status stored by the IPL termination may be to a memory location other than the standard location 0.

Figure 2-5 illustrates the format of the MPX-32 operating system system distribution tape (SDT). The tape is designed to operate with any standard class E or class F magnetic tape channel and is loaded by the IPL function.

Figure 2-6 illustrates the format of the CONCEPT/32 diagnostic tape/disc distribution facility. The diagnostic facility software is loaded by an IPL function.

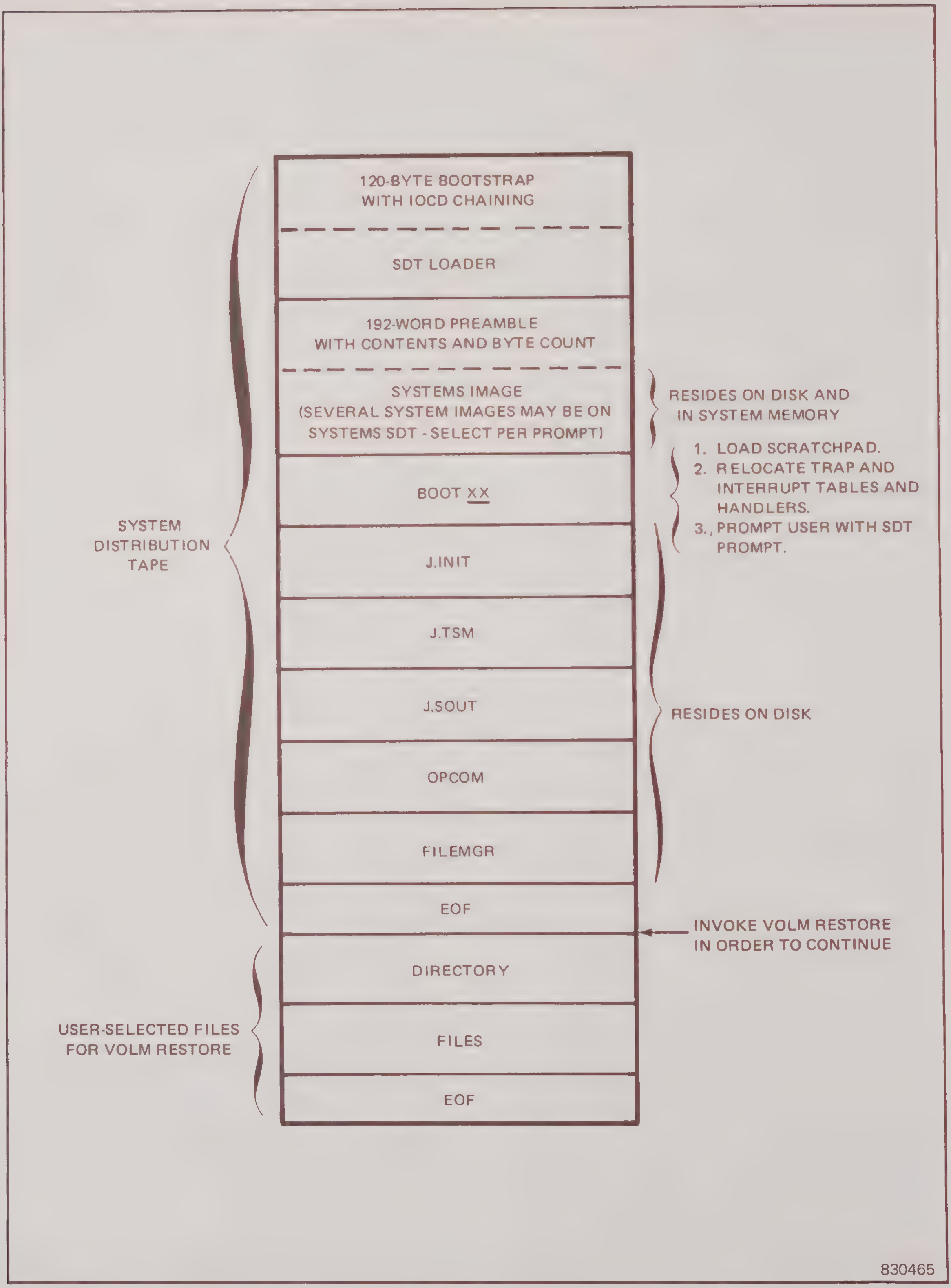

120-BYTE BOOTSTRAP WITH IOCD CHAINING
SDT LOADER
192-WORD PREAMBLE WITH CONTENTS AND BYTE COUNT
SYSTEMS IMAGE (SEVERAL SYSTEM IMAGES MAY BE ON SYSTEMS SDT - SELECT PER PROMPT)
BOOT XX
J.INIT
J.TSM
J.SOUT
OPCOM
FILEMGR
EOF
DIRECTORY
FILES
EOF
SYSTEM DISTRIBUTION TAPE
USER-SELECTED FILES FOR VOLM RESTORE
RESIDES ON DISK AND IN SYSTEM MEMORY
1. LOAD SCRATCHPAD.
2. RELOCATE TRAP AND INTERRUPT TABLES AND HANDLERS.
3., PROMPT USER WITH SDT PROMPT.
RESIDES ON DISK
INVOKE VOLM RESTORE IN ORDER TO CONTINUE
830465

Figure 2-5. MPX-32 System Distribution Tape Format

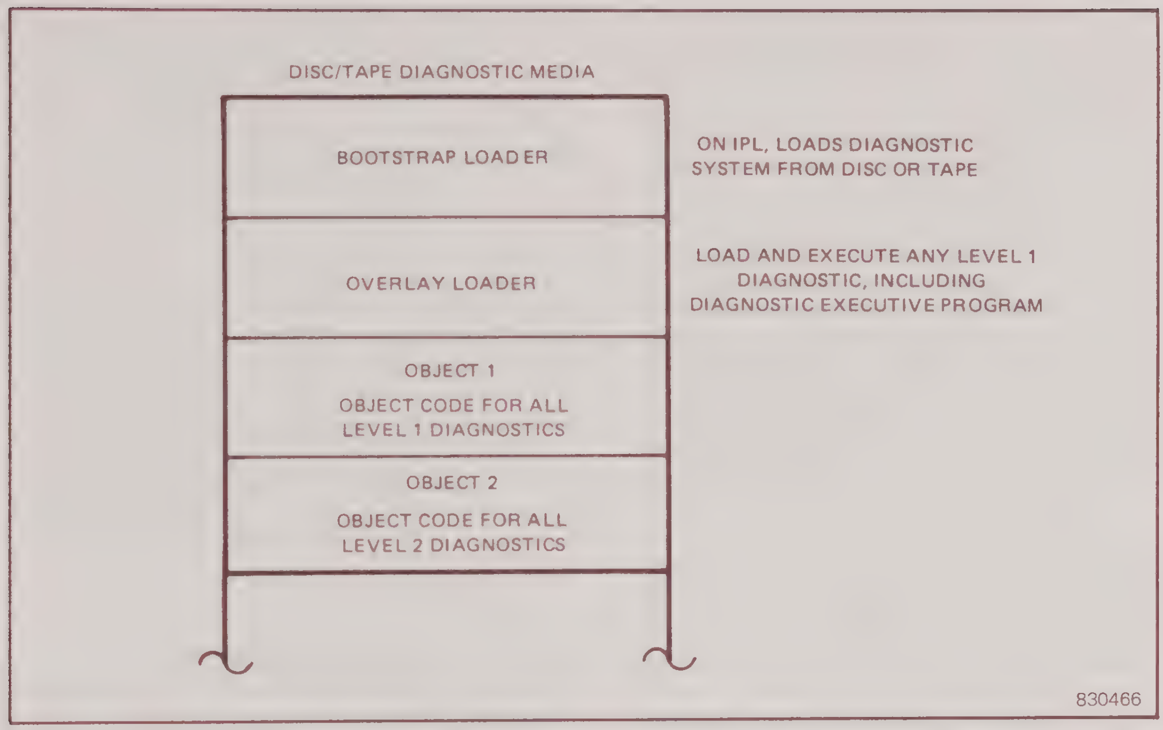

DISC/TAPE DIAGNOSTIC MEDIA
BOOTSTRAP LOADER
ON IPL, LOADS DIAGNOSTIC SYSTEM FROM DISC OR TAPE
OVERLAY LOADER
LOAD AND EXECUTE ANY LEVEL 1 DIAGNOSTIC, INCLUDING DIAGNOSTIC EXECUTIVE PROGRAM
OBJECT 1
OBJECT CODE FOR ALL LEVEL 1 DIAGNOSTICS
OBJECT 2
OBJECT CODE FOR ALL LEVEL 2 DIAGNOSTICS
830466

Figure 2-6. Diagnostic Tape/Disc Distribution Facility Format

2.5.2.7 IPL Termination

IPL termination is defined as the point in time when the IPL data transfer from the IPL media to memory is completed. IPL termination under class E protocol is determined differently than under class F protocol.

In class E protocol, the CPU will cycle TD8000 SelBUS communication transfers while the IPL data transfer is in progress. For the duration of the data transfer, the class E channel responds to the TD8000 communication sequence with an I/O channel busy indication. When the I/O channel no longer responds with the busy indication, IPL termination has occurred. A busy indication can be any one of the following responses:

1. SelBUS I/O retry signal
2. SelBUS channel busy signal
3. Bit 16 true in a SelBUS TD8000 data return transfer.

Class E protocol has no provisions for an IPL termination status transfer; therefore, the CPU cannot determine if I/O errors occurred during the IPL sequence. When IPL termination occurs the CPU enters the fill pipeline sequence and attempts to execute the IPL data starting at memory location 0.

In class F protocol, IPL termination is identified by the occurrence of a pseudo interrupt, generated by the IPL channel and sent to the CPU. The CPU responds to the pseudo interrupt with a SelBUS acknowledge and deactivate communication sequence.

When the I/O channel receives the acknowledge and deactivate command, it stores a termination status doubleword into memory locations 0 and 4. The I/O channel then sends to the CPU a status stored condition code configuration and the memory address of the termination status doubleword. The condition code configuration and status memory address are sent to the CPU in a data return transfer as part of the SelBUS acknowledge and deactivate communication sequence.

Note that if the IPL input/output command list contained a channel initialization (INCH) command, the termination status will be stored in memory locations other than 0 and 4. However, the memory address of the termination status doubleword is communicated to the CPU as part of the class F protocol. Unfortunately, the termination status address is not stored in memory, and is therefore, not software apparent.

The format of the class F channel status doubleword is provided with the class F I/O description in the 32/67 CPU reference manual. Essentially, the first word of the doubleword pair contains the next IOCD address, and the second word contains the termination status flags in bits 0 through 15.

When IPL termination occurs, the CPU examines the termination flags to determine if the IPL data transfer was error free. If no I/O errors occurred, the CPU will complete the IPL sequence. If errors are present, the CPU executes an automatic trap halt.

In the termination flags (word 2, bits 0 through 15) of the termination status doubleword, only bits 12 and 13, device end and channel end, should be set. The following termination flag bits must be false:

1. Bit 3, channel program check
2. Bit 4, channel data check
3. Bit 5, channel control check
4. Bit 6, interface check
5. Bit 7, chaining check
6. Bit 8, busy
7. Bit 9, status modifier
8. Bit 10, attention
9. Bit 14, unit check
10. Bit 15, unit exception

Normally, the termination status doubleword is stored in memory locations 0 and 4, overlaying the CPU program status doubleword, read from the first IPL record. This overlay presents no problem since the CPU reads the PSD from memory locations 0 and 4 and stores the PSD internally prior to IPL termination.

2.5.2.8 CPU Fill Pipeline

CPU fill pipeline refers to that portion of the IPL sequence which sets the operating state of the CPU, fills the instruction pipeline, and passes control to the software bootstrap program. The fill pipeline sequence is different for class E and class F protocols. If IPL errors have been previously detected, the fill pipeline sequence is not executed, but instead an automatic trap halt is executed.

In class E rules the operational state of the CPU is established by the CPU firmware as follows:

1. The CPU is placed in the run state.
2. The contents of the software GPRs are unchanged.
3. All I/O devices and interrupt devices have been reset.
4. The CPU is privileged.
5. The condition codes are zero.
6. Extended addressing is off.
7. Base mode is off.
8. Arithmetic exception is disabled.
9. Macro program counter is set to zero.
10. Mapped environment is off.
11. The map CPIX and BPIX fields are set to zero.
12. Interrupts are unblocked.

13. The effective program status doubleword is
    @8000 0000 PSD word 1
    @0000 0000 PSD word 2
14. Software instruction begins at memory location 0.

In class F rules the operational state of the 32/67 CPU is established by the IPL program status doubleword and the CPU firmware as follows:

1. The CPU is placed in the run state.
2. The contents of the software GPRs are unchanged.
3. All I/O devices and interrupt devices have been reset.
4. The CPU privileged state is determined by IPL PSD word 1.
5. The condition codes are determined by IPL PSD word 1.
6. Extended addressing is determined by IPL PSD word 1.
7. Base mode is determined by IPL PSD word 1.
8. Arithmetic exception enable/disable is determined by IPL PSD word 1.
9. The value of the macro program counter is determined by IPL PSD word 1.
10. Mapped environment is off.
11. The map CPIX and BPIX fields are set to zero.
12. Interrupts are blocked.
13. The effective program status doubleword is:
    IPL PSD1 : PSD word 1
    @0000 4000 : PSD word 2
14. Software instruction begins as determined by IPL PSD word 1.

2.6 Powerdown Sequence

The power-down sequence consists of a hardware/firmware sequence that is designed to detect a SelBUS power-fail signal and then transfer control to the software power-down trap handler. The purpose of the power-down trap is to provide the required parameters for a subsequent power-up auto restart sequence when power is restored to the system. Figure 2-7 provides a flowchart of the power-down sequence.

The power-down hardware consists of the power-fail detection hardware and the corresponding 32/67 CPU hardware. The power-fail detection hardware senses a pending power failure and generates the SelBUS power-fail signals. The CPU/IPU hardware detects the SelBUS power-fail signal and translates the SelBUS signal into a firmware event that can be scheduled. The SelBUS power-fail signals consist of three signals that occur in a timed sequence as follows:

1. The power-fail (LPF) signal occurs first, indicating that approximately two milliseconds of reliable power remain.
2. The reset (LRESET) signal occurs approximately two milliseconds after the power-fail signal. The LRESET signal forces the CPU and all devices connected to the SelBUS to a reset state.
3. The power-fail memory (LPFM) signal occurs approximately ten microseconds after the reset signal. The LPFM signal protects the memory from inadvertent signals during the actual power down.

The CPU/IPU hardware detects the SelBUS power-fail signal and immediately generates a power-fail event signal to the firmware which will schedule the power-fail event. The power-fail event is scheduled at the following points:

1. Between instructions

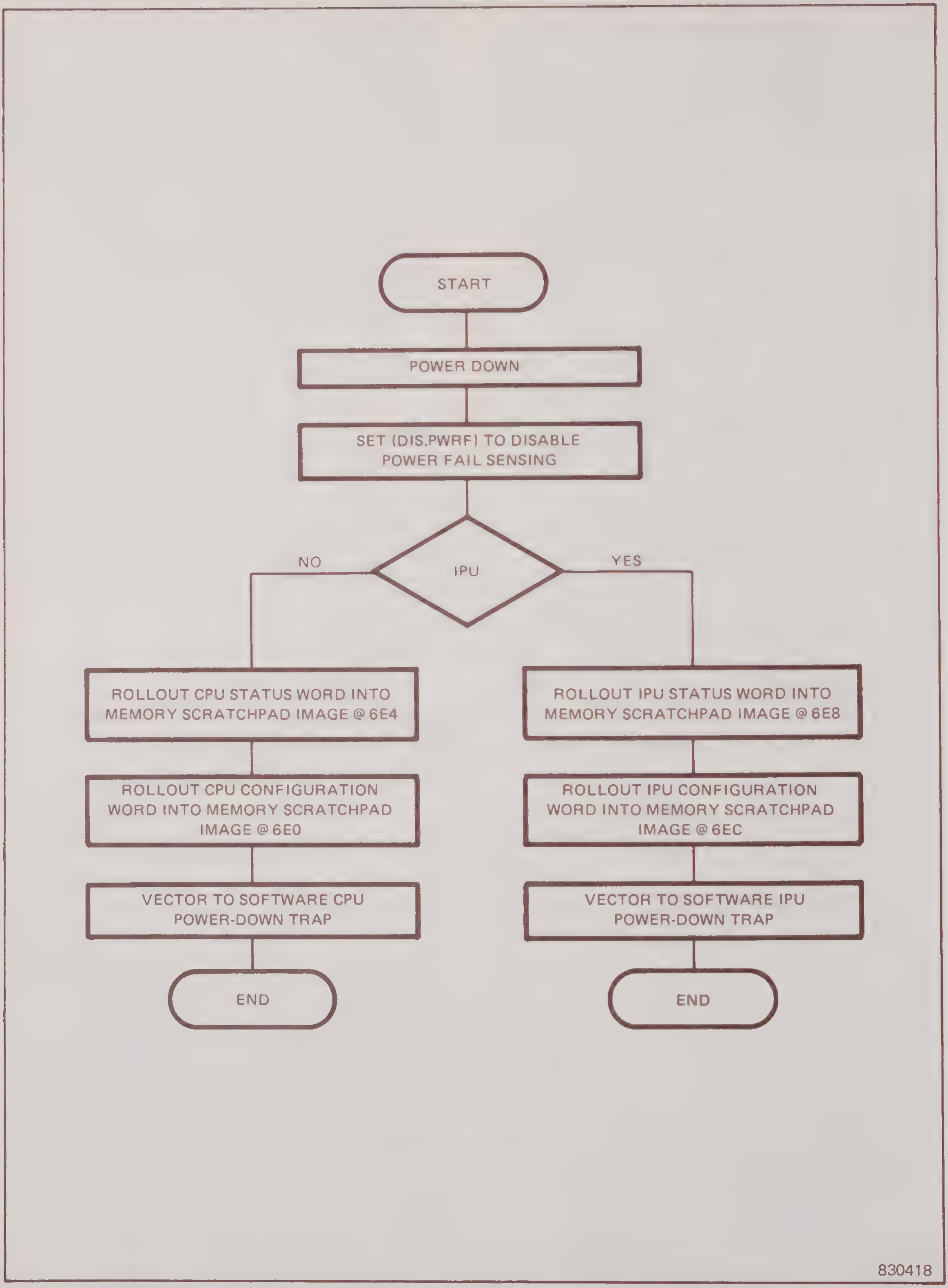

START
POWER DOWN
SET (DIS.PWRF) TO DISABLE POWER FAIL SENSING
IPU
NO
YES
ROLLOUT CPU STATUS WORD INTO MEMORY SCRATCHPAD IMAGE @ 6E4
ROLLOUT CPU CONFIGURATION WORD INTO MEMORY SCRATCHPAD IMAGE @ 6E0
VECTOR TO SOFTWARE CPU POWER-DOWN TRAP
END
ROLLOUT IPU STATUS WORD INTO MEMORY SCRATCHPAD IMAGE @ 6E8
ROLLOUT IPU CONFIGURATION WORD INTO MEMORY SCRATCHPAD IMAGE @ 6EC
VECTOR TO SOFTWARE IPU POWER-DOWN TRAP
END
830418

Figure 2-7. Powerdown Sequencing Flowchart

2. During an instruction sequence, or an interrupt-I/O sequence, as follows:

   a. After every indirect fetch that produces a subsequent indirect fetch

   b. After a SelBUS advance transfer and while the CPU is waiting for ready (I/O or interrupt operations)

   c. At certain predefined points in long instruction sequences, such as firmware floating-point.

3. While the CPU is halted.

Note that the power-fail event is not scheduled in the following sequences:

1. While the CPU is waiting for an I/O or memory DRT

2. Following the last indirect fetch of an indirect chain

3. Between doubleword fetches or doubleword instruction stores

4. Between multiple fetches/stores executed in the load/store file instructions.

If a power fail occurs while the CPU/IPU has halted, the corresponding bit in the status word will be set in the CPU/IPU status word. The CPU status word indications are provided in table 2-6.

The CPU/IPU trap status words and configuration words will be stored in the power-down trap context block during the power-down sequence. The CPU and IPU scratchpad keywords are not rolled out to memory during power down; they must be placed in the memory scratchpad image by software for a successful auto restart.

It becomes a software responsibility to determine if the power-fail interrupted process can be restarted via the software power-up trap.

After the CPU/IPU has detected and scheduled the power fail, it stores the current CPU/IPU status words in the memory scratchpad image. The scratchpad image status word is used by the power-up auto restart sequence to determine if traps were enabled during the power-down sequence. If traps were enabled, the firmware disables subsequent power-fail detection and executes a standard trap to the power-down trap handler. If traps were not enabled, the CPU/IPU executes an automatic trap halt. Power-fail detection remains disabled until software executes either a load program status doubleword (LPSD) or load program status doubleword and change map (LPSDCM) instruction. Power-up software must also use the status words and configuration words to reconfigure the CPU/IPU to the same configurations that were in effect prior to the power fail.

Assuming that traps were enabled when the software power-down trap handler receives control of the CPU, a minimum of 1.5 milliseconds exists before instruction execution is stopped by the reset signal. In this time interval, software must store the general-purpose registers and base registers into memory to preserve their contents during the power outage.

During the power-down sequence, the CPU rolls out only the current CPU status word to memory location 6E4 and the CPU configuration word to memory location 6E0. The IPU rolls out only the current IPU status word to memory location 6E8 and the IPU configuration word to memory location 6EC.

Scratchpad memory image locations 6D4, 6E0, 6E8, and 6EC are normally defined for alternate uses. However, during power-down sequences, these image locations are redefined by CPU/IPU firmware to provide the functionality required for the auto-restart feature. Table 2-7 lists these scratchpad redefinitions.

Table 2-6
CPU Read Status Word Format

Bit	Status
Bit 00 = 0	Unprivileged mode
= 1	Privileged mode
01-04 =	Not used
05 = 0	Extended addressing disabled
= 1	Extended addressing enabled
06 = 0	Base register mode disabled
= 1	Base register mode enabled
07 = 0	Arithmetic exception trap disabled
= 1	Arithmetic exception trap enabled
08 = 0	Map disabled
= 1	Map enabled
09-19 =	Not used
20 = 0	Write to ACS and WCS disabled
= 1	Writes to ACS and WCS enabled
21 = 0	PROM mode enabled
= 1	ACS mode enabled
22 = 0	FPA present and enabled
= 1	FPA not present
23 = 0	Privileged mode halt trap disabled
= 1	Privileged mode halt trap enabled
24 = 0	Interrupts are unblocked
= 1	Interrupts are blocked
25 = 0	Software traps are disabled (Automatic trap halt is enabled)
= 1	Software traps are enabled
26 =	Not used
27 = 0	CPU configuration
= 1	IPU configuration
28-31 =	CPU model indicator
0000 =	32/55 CPU
0001 =	32/75 CPU
0010 =	32/27 CPU
0011 =	32/67 CPU
0100 =	32/87 CPU
0101 =	32/97 CPU
0110 =	DP 67 CPU
0111-1111 =	Not defined (reserved)

2.7 Internal Processing Unit (IPU)

The CONCEPT 32/6780 contains an IPU. An IPU consists of a second set of CPU boards. This second processor is configured as an IPU by the turnkey panel PROCESSOR SELECT switch and processor identity jumpers located on the two CS Units (refer to the 32/67 System Manual for jumper locations). At the hardware level, the two processors are identical; they share the same SelBUS and memory. The IPU provides the facility to offload tasks from the primary CPU, thereby increasing the computational performance of the system. Although

Table 2-7
Power Up/Down Scratchpad
Memory Image Location Redefinition

Memory Image Address	Power Up/Down Redefinition and Address		Normal Scratchpad Definition and Address	
	Definition	Address	Definition	Address
6D4	IPU Keyword X'13254768'	IPU - F7	Current PSD2	F5
6E0	CPU Configuration Word	CPU - 1FE	Identify Device Protocol DRT	F8
6E8	IPU Status Word	IPU - F9	Current Active Interrupt	FA
6EC	IPU Configuration Word	IPU - 1FE	Number of Active Interrupts	FB

both processors share the SelBUS, they cannot share I/O channels that plug into it since SelBUS protocol does not provide for I/O channel sharing between multiple processors. The optional IPU Console provides the IPU with panel capabilities and console I/O capabilities.

The following text is subdivided into IPU operation, IPU characteristics, and IPU console.

2.7.1 IPU Operation

Both the CPU and IPU processors share a common SelBUS. However, they do not communicate with each other via SelBUS transfers. One processor may gain the attention of the other by using one of two unidirectional SelBUS lines designated SIGNAL IPU and SIGNAL CPU. Full details on all SelBUS lines are contained in the 32/67 System Manual.

The SIGNAL IPU SelBUS line originates in the CPU and is received by the IPU. This line is driven by the CPU when software executing in the CPU uses the signal IPU (SIPU) instruction. When this signal is received by the IPU, it causes the IPU to execute a CPU/IPU trap (assuming traps and interrupts are unblocked in the IPU). Software residing in the CPU/IPU trap handler of the IPU must define the meaning and purpose of the CPU/IPU trap.

Similarly, the SIGNAL CPU SelBUS line originates in the IPU and is received by the CPU. When software residing in the IPU uses the SIPU instruction, the SIGNAL CPU line is driven by the IPU. When this signal is received by the CPU, it causes the CPU to execute an IPU/CPU trap (assuming traps and interrupts are unblocked in the CPU). Software residing in the IPU/CPU trap handler of the CPU must define the meaning and purpose of the IPU/CPU trap. The IPU and CPU trap handler software can communicate via their common memory.

In both the CPU and the IPU, if a CPU/IPU or IPU/CPU trap occurs while interrupts are blocked, the trap will become pending and will remain pending until traps are unblocked. If a second trap occurs while a first trap is pending, the second trap is lost. However, the first trap remains pending until traps are unblocked.

As far as the operating system is concerned, the CPU and the IPU are totally symmetrical processors. Therefore, when using the IPU, the operating system takes an entire task and places it in the IPU for execution. The task executes in the IPU until: the task requests an operating system service that cannot be executed in the IPU (software restriction), executes an IPU restricted instruction (hardware restriction), generates a hardware/-software error, or reaches normal completion.

During the time that the IPU is executing a task, the CPU is free to resume task scheduling or a second task. When the task terminates in the IPU, the IPU software signals the CPU and CPU software must then determine the reason for the termination. If the cause of termination is an error, the task will be aborted. If the cause of termination is a request for an IPU restricted service or instruction, the CPU performs that service or instruction on behalf of the IPU. The CPU then returns control of the task to the IPU.

Since the operating system uses the IPU at task level, the performance of a single task is not improved by using the IPU. However, when considered in a multi-task environment, the total system throughput is increased by the IPU since two tasks may execute simultaneously (one in the CPU and one in the IPU).

In order to comply with the software concept, the CPU and IPU must be symmetrical in that they have the same options and produce the same results. However, a notable exception is that the IPU does not have the ability to perform system level I/O operation. This is due to both IPU hardware and software, which causes operating system calls for I/O activity or I/O instructions to be trapped back to the CPU. The CPU then performs the I/O activity for the IPU (task).

Both processors (CPU and IPU) must be defined with the same options and features at system configuration time. These include alterable control store (ACS), writable control store (WCS), and floating-point accelerator (FPA).

At system initialization time, software must not enable a feature or option that is not available or is inoperable in the other processor.

2.7.2 IPU Operational Characteristics

The IPU operational characteristics are defined by the IPU hardware and firmware sequences which differentiate IPU execution from CPU execution. The following paragraphs define and describe these differences.

The IPU is provided with a full set of error detection traps plus one additional trap that does not exist in the CPU. This additional trap is the IPU undefined instruction trap, caused by the block external interrupt (BEI) instruction. I/O instruction operation codes are defined as valid or invalid for the CPU or IPU. This information is contained in table 2-8. Full details of I/O classes are provided in later paragraphs under the heading of I/O Class Identification.

To differentiate IPU traps from CPU traps in the common memory, the IPU trap table base address in scratchpad location F0 is set to a different value than that used for the CPU trap table base address. Normally, the IPU trap table base address is set to 20, while the CPU trap table base address is set to 80. Table 2-9 lists the CPU and IPU traps and their default trap vector locations. Since traps are always enabled in the IPU, CPU software must initialize all IPU trap vector locations before initializing the IPU.

I/O and interrupt instructions, are not catagorically undefined in the IPU. However, they are rejected as IPU undefined when used with standard classes of I/O such as class F, 3, and E.

Table 2-8
IPU and CPU I/O Classes

I/O Class	CPU	IPU	Description
3	Valid	Undefined IPU Trap	CPU Interval Timer
E	Valid	Undefined IPU Trap	CPU Real-time Clock and External Interrupts
F	Valid	Undefined IPU Trap	CPU IOP Channel Devices
B	Invalid. System Check Trap	Valid	IPU Interval Timer
6	Invalid. System Check Trap	Valid	IPU Real-time Clock and External Interrupts
7	Invalid. System Check Trap	Valid	IPU IOP Channel Devices

Table 2-9
IPU and CPU Trap Vector Locations

Trap Number (Hex)	CPU Trap Vector (Hex)	IPU Trap Vector (Hex)	Definition
00	80	20	Power fail trap
01	84	24	Poweron trap
02	88	28	Memory parity trap
03	8C	2C	Nonpresent memory trap
04	90	30	Undefined instruction trap
05	94	34	Privilege violation trap
06	98	38	Supervisor call trap
07	9C	3C	Machine check trap
08	A0	40	System check trap
09	A4	44	MAP fault trap
0A	NU	48	Undefined IPU instruction trap
0B	AC	4C	Start IPU or IPU finished
0C	B0	50	Address specification error
0D	B4	54	Console attention
0E	B8	58	Privilege mode halt trap
0F	BC	5C	Arithmetic exception
10	C0	60	Cache fault

More specifically, extended I/O instructions such as ECI, DCI, SIO, etc., cannot be executed to class F devices but they may be executed to class 7 devices. Command device (CD) and test devices (TD) instructions cannot be executed to class 3 or E (D) devices, but they may be executed to class B devices. Interrupt control (EI, DI, etc.) instruction cannot be executed to class E devices, but they can be executed to class 6 devices. The new I/O and interrupt classes (6, 7, and B) are used to define I/O and interrupts dedicated to the IPU, and these devices exist only with the IPU console option (described in later paragraphs).

The standard I/O and interrupt device classes (3, E, and F) are used to define I/O and interrupts dedicated to the CPU. If the CPU attempts to access a device defined for the IPU, CPU firmware causes a CPU trap.

Normally, the IPU and CPU can use the same base address for the interrupt table and master process list.

The IPU is provided with a different scratchpad keyword from the CPU. The different keyword ensures that firmware can determine the correct scratchpad default values when CPU and IPU roles are reversed by means of the turnkey panel PROCESSOR SELECT switch and the execution of a system reset.

As is the case in the CPU, the presence of the correct keyword in the scratchpad inhibits the firmware from establishing default values for the scratchpad table base addresses and defaulting all options and features.

At system configuration time, one common scratchpad image can be developed for the CPU and IPU if:

1. The IPU console option is used.

2. Software I/O channel addresses and interrupt levels are chosen such that the software addresses are mutually exclusive of the CPU I/O software channel addresses and interrupts.

The exceptions to the commonality between the scratchpad image are the IPU trap table base address and keyword. Handling these as exceptional cases in software causes no major problems to the IPU hardware and firmware, since the IPU does not automatically roll in the scratchpad image.

At power-down time, the IPU rolls out only two scratchpad locations. These are the IPU status word and the IPU configuration word. The memory locations used for the power-down rollout are located in the CPU scratchpad memory image, and are locations in that image that are not required by the CPU for auto restart at powerup.

At powerup, the IPU firmware checks a location in the CPU memory scratchpad image for the presence of the IPU keyword. If the keyword is present, the IPU scratchpad is cleared, all options and features are defaulted, and the trap table base address is set to 020. At this point, the IPU enters the IPU power-up trap sequence for auto restart, and software must complete the initialization. During powerup, no scratchpad locations are rolled into the IPU.

At powerup, if the IPU keyword is not present in the CPU memory scratchpad image, the IPU clears its scratchpad base addresses and defaults all options and features. It then enters the halt state and firmware idle loop to wait for a CPU/IPU trap.

2.7.3 IPU Console

IPU I/O capability is provided by the optional IPU Console. Basically, this option comprises an IOP, CRT, and cabling. The IOP firmware and hardware is designed to interface with different SelBUS lines and the IPU I/O SelBUS interface. The IPU Console is dedicated for operation with the IPU and cannot be accessed by the CPU. Similarly, all other SelBUS I/O channels are dedicated to CPU operation and cannot be accessed by the IPU.

The primary purpose of the IPU Console is to provide the IPU with panel function capabilities. It provides all the panel functions that are available to the CPU except for IPL and control of clock override. With this option, it is possible for IPU software to use the I/O console device, the real-time clock, the interval timer, and external interrupt levels provided by this IPU Console IOP.

IPU software uses standard I/O and interrupt control instructions to control I/O and external interrupt levels. IPU firmware then uses the contents of the IPU scratchpad device and interrupt entries to determine if the I/O device (channel) or interrupt level specified can be operated by the IPU. Specifically, the I/O class field of the device and interrupt entry is used to determine if the I/O device/interrupt can be controlled by the IPU.

2.7.3.1 IPU Console Configuration

When configuring the option into the CPU/IPU system, all of the physical addresses, channel addresses, device addresses, and interrupt levels assigned and dedicated to the CPU must be known. The IPU Console is then assigned to a physical address with channel addresses, device addresses, and interrupt levels that are unique to it. This technique allows the development of a single scratchpad image for the CPU and IPU with the only IPU differences being the IPU trap table base address and the IPU keyword.

It is feasible to assign software channel addresses, device addresses, and interrupt levels to the IPU Console that are also used in the CPU to describe different channels and devices. However, this would require the development and maintenance of two separate scratchpad images; one for the CPU, the other for the IPU. The interrupt table base address would also have to be different from the IPU interrupt table base address.

The recommended technique for configuring the IPU Console is to use unique channel addresses, device addresses, and interrupt levels with one scratchpad for the CPU/IPU pair.

The following paragraphs describe specific parameters for configuring the IPU Console.

2.7.3.1.1 Physical Address

The IPU Console must be configured at a unique physical address occupying an odd and even address pair to provide channel and interrupt functionality. To be used by the IPU, the IOP must have a physical address that is divisable by four. As an example, if physical address 74 is used, when communicating to the panel functions of the IOP, the IPU firmware uses physical address 02. For the real-time clock, interval timer, and external interrupt functions of the IOP, the odd physical address variation of the IOP standard physical address must be used (e.g., address 75).

2.7.3.1.2 Channel Address

The IPU Console channel address is made up of the software 7-bit address of the channel and the IPU scratchpad address for that channel device entry. This channel address should be unique when compared with the CPU channel addresses. Normally, the channel address will match the IPU Console physical address. This facilitates the I/O software address to physical address mapping scheme (provided by the IPU scratchpad).

The channel address must be used with the extended I/O instructions to communicate with the I/O channel or its interrupt level. When the channel address is appended to the 8-bit subaddress, software can use the channel and subaddress with the extended I/O instructions to communicate with IPU I/O device (console, floppy disc, etc.).

2.7.3.1.3 Channel Subaddress

Channel subaddresses and device subaddresses should be assigned to conform with standard IOP configurations. In many cases, such as console receive and transmit functions, the channel and device subaddresses are dedicated to specific device functions. Standard subaddress assignments should be used whenever possible to avoid potential confusion between the CPU IOP and the IPU Console IOP configurations.

2.7.3.1.4 Channel Interrupt Level

The IPU Console interrupt level should be assigned to a unique level within the range 00 to 6F. The channel interrupt value plus 80 provides the hexadecimal address in the IPU scratchpad of the IPU Console channel interrupt entry.

The channel interrupt level (when used as a word pointer, or index, and added to the interrupt table base address obtained from the IPU scratchpad) provides the interrupt vector location (IVL) for the IPU Console IOP channel interrupt. Software must initialize the IVL and provide an interrupt context block (ICB) for the IPU Console channel interrupt in the same manner as for a CPU IOP channel interrupt. The ICB must be formatted according to standard extended I/O (class F) protocol.

2.7.3.1.5 Interval Timer Device Address

The interval timer in the IPU Console IOP should be assigned to a unique software device address. The Software device address provides the address for IPU software command device (CD) and test device (TD) instructions to communicate with the interval timer in the IPU Console IOP. The software device address also provides the address in the IPU scratchpad of the IPU Console IOP interval timer device entry. The device entry serves as the mapping function to convert the software device address into a SelBUS physical address and subaddress of the interval timer in the IPU Console IOP. The device entry must contain the odd address variation of the IPU Console IOP physical address pair.

2.7.3.1.6 Interval Timer Interrupt Level

The interval timer in the IPU Console IOP should be assigned to a unique interrupt level. The software interrupt level provides the interrupt level for the software interrupt control instructions (EI, DI, etc.) to communicate with the interval timer in the IPU Console IOP. The software interrupt level, with 80 added, provides the interval timer interrupt entry. The interrupt level must be in the range 00 to 6F.

The software interrupt level (when used as a word pointer, or index, and added to the interrupt table base address obtained from the IPU scratchpad) provides the IVL for the interrupt timer. Software must initialize the IVL and provide an ICB for the interval timer in the IPU Console IOP.

The IPU scratchpad interrupt entry provides the physical and subaddress of the interval timer in the IPU Console IOP. The physical address must be the odd variation of the IPU Console IOP address pair.

2.7.3.1.7 Real-time Clock Interrupt Level

The IPU Console IOP real-time clock interrupt level functions as any other IPU Console IOP interrupt level with the exception that this level automatically requests interrupts at an ac line frequency rate if the level is enabled.

The IPU Console IOP real-time clock should be assigned to a unique software interrupt level. The software interrupt level provides the interrupt level for software interrupt control instructions (EI, DI, etc.) to communicate with the IPU Console IOP real-time clock interrupt level. The software interrupt level, when added to 80, provides the IPU scratchpad address of the IPU Console IOP real-time clock interrupt entry.

The software interrupt level (when used as a word pointer, or index, and added to the interrupt table base address obtained from the IPU scratchpad) provides the IVL for the IPU Console IOP real-time clock. Software must initialize the IVL and provide an ICB for the IPU Console IOP real-time clock.

The scratchpad interrupt entry provides the physical and subaddress of the IPU Console IOP real-time clock. The physical address must be the odd variation of the IPU Console IOP physical address pair.

2.7.3.1.8 External Interrupt Levels

The IPU Console IOP external interrupts should be assigned unique software interrupt levels. These software interrupt levels function as previously described for the real-time clock interrupt with the exception that the external levels do not automatically request an interrupt, if enabled.

2.7.3.2 I/O Class Identification

The I/O instructions used to control the CPU and IPU I/O devices and interrupts are identical. To differentiate between those devices controlled by the CPU and those controlled by the IPU, the IPU scratchpad device and interrupt entries use I/O classes that are subsets of the CPU I/O and interrupt classes.

The following paragraphs describe the IPU I/O classes as subsets of the corresponding standard CPU I/O classes. In all cases, the subset I/O classes only relate to those I/O devices and interrupts that are controlled by the IPU Console IOP. The subset I/O class identification value is obtained by taking the standard class field from the device and interrupt scratchpad entries and ones-complementing the most significant bit of the 4-bit class field. Therefore, class 7 is the subset of class F, class B is the subset of class 3, and class 6 is the subset of class E.

2.7.3.2.1 Class 7

Class 7 I/O describes the IPU Console IOP channel devices. Each of these channels and devices obey class F (extended I/O) protocol rules. Class 7 is the subset of class F. Class 7 is defined only in the IPU, and must be used in scratchpad channel (device) and interrupt entries to specify the IPU Console IOP channel and devices.

2.7.3.2.2 Class B

Class B I/O describes the IPU Console IOP interval timer and its associated interrupt level. Class B is the subset of class 3, and the IPU Console IOP interval timer obeys class 3 (CD/TD) protocol rules.

Class B must be used in the device and interrupt scratchpad entries to describe the IPU Console IOP interval timer.

2.7.3.2.3 Class 6

Class 6 describes the IPU Console IOP real-time clock interrupt and external interrupts. Class 6 I/O identifies those interrupt levels that obey class E interrupt protocol (EI, DI, etc.) but are dedicated to operation with the IPU Console IOP. The real-time clock and external interrupts are classified as non-I/O interrupts in both the CPU and IPU.

In the CPU, non-I/O entries have a class field of zero. However, the class field is not verified. In the IPU, non-I/O interrupts must have a class field equal to 6 to indicate validation for IPU operation.

The external interrupts (non-I/O) have only scratchpad interrupt entries (no device entries). The entry is defined as non-I/O by bit 8 equal to one.

2.8 Automatic Trap Halt

The automatic trap halt feature is a firmware sequence designed to report traps to an operator when software cannot handle the trap. Automatic trap halts can occur in one of the following conditions:

1. When software traps are disabled
2. When there are memory errors during powerup auto restart
3. When there are memory errors or I/O errors during IPL or auto IPL.
4. If the firmware detects an error while in a firmware trap handler, and the trap handler is trying to report the same type of trap error (recursive trap error).

Automatic trap halts are not used for the arithmetic exception trap or a supervisor call. If an arithmetic exception occurs while traps are disabled, it is ignored. A supervisor call is handled normally. Automatic trap halt indications are as follows.

1. The CPU/IPU is halted.
2. The interrupt active indicator is on; however, no interrupts are active.
3. The PSD word 1 display has indicated the dedicated trap vector address in the program counter. In some cases, the halfword flag is on but should be ignored.
4. Memory location 680 contains PSD word 1 at the time of the trap (690 for IPU).
5. Memory location 684 contains PSD word 2 at the time of the trap (694 for IPU).
6. Memory location 688 contains the CPU trap status word (698 for IPU).
7. Memory location 68C contains the contents of R(DVC) which is either a scratchpad device entry or an interrupt entry and is valid only if the old PSD word 1 and the CPU trap status word indicate that an I/O or interrupt sequence was in progress (69C for IPU).

The IPU normally has traps enabled and, therefore, does not normally execute automatic trap halts. In the case of software disabling IPU traps, memory locations are provided for IPU trap halt indicators.

2.9 CPU/IPU Scratchpad

A 256-location by 32-bit scratchpad memory is provided in the CPU/IPU hardware. The CPU/IPU scratchpad provides an emulation work area, storage for the I/O device entries, interrupt structure information, and various base addresses.

The I/O device entries and interrupt entries are part of the initial configuration list (ICL) which is loaded during initialization. (The system configuration information remains in the processor scratchpad as long as power is applied to the processor.) I/O device entries are used to map software device addresses into SelBUS physical addresses. Interrupt entries are used to map software interrupt levels into SelBUS physical addresses.

An allocation table for the scratchpad is provided in figure 2-8. The allocation table also indicates the memory address of the memory scratchpad image.

Software is responsible for loading both CPU and IPU scratchpads during system initialization. Only scratchpad locations 00 thru F4 and location F7 (keyword) must be loaded. The remaining scratchpad locations are initialized by the system reset functions of the processors. Scratchpad locations F0 (trap table base address) and F7 (keyword) are the only locations that must be different in the CPU and IPU scratchpads.

2.9.1 Scratchpad Image

Normally only one scratchpad image is required for CPU/IPU operation, with the exception of the IPU trap table base

CONCEPT 32/67 CPU/IPU SCRATCHPAD

MEMORY IMAGE ADDRESS	SCRATCH-PAD ADDRESS		DECIMAL
300	0	CHANNEL - DEVICE 0	0
↓	↓	DEVICE ENTRIES	↓
4FC	7F	CHANNEL - DEVICE 7F	127
500	80	INTERRUPT LEVEL 0	0
↓	↓	INTERRUPT ENTRIES	↓
6BC	EF	INTERRUPT LEVEL 6F	111
6C0	F0	TRAP TABLE BASE ADDRESS (CPU = 080, IPU = 020)	NOTES 1, 2
6C4	F1	INTERRUPT TABLE BASE ADDRESS (= 100)	NOTES 1, 3
6C8	F2	IOCD BASE ADDRESS (CLASS E) (= 700)	NOTES 1, 3
6CC	F3	MASTER PROCESS LIST (MPL) BASE ADDRESS (= 788)	NOTES 1, 2
6D0	F4	DEFAULT IPL ADDRESS	
6D4	F5	CURRENT PSD2	
6D8	F6	RESERVED	
6DC	F7	SCRATCHPAD KEY (CPU = X'ECDAB897') (IPU = X'13254768')	
6E0	F8	IDENTIFY DEVICE PROTOCOL DRT	
6E4	F9	CPU STATUS WORD	
6E8	FA	CURRENT ACTIVE INTERRUPT	
6EC	FB	NUMBER OF ACTIVE INTERRUPTS	
6F0	FC	AUTO IPL DEVICE ADDRESS (= 0 IN MANUAL IPL)	
6F4	FD	RESERVED	
6F8	FE	RESERVED	
6FC	FF	INTERRUPT LEVEL 7F = 00FFFFF	

NOTE 1. DENOTES LOCATIONS THAT MUST BE PROVIDED BY THE SOFTWARE FOR ICL.

NOTE 2. THE TRAP TABLE AND THE MASTER PROCESS LIST MUST RESIDE IN THE FIRST 128K WORDS OF MAIN MEMORY.

NOTE 3. THE INTERRUPT TABLE AND INPUT/OUTPUT COMMAND DOUBLEWORD MUST RESIDE IN THE FIRST 128K WORDS OF MAIN MEMORY.

830421

Figure 2-8. Allocation of CPU/IPU Scratchpad

address and the IPU keyword. Software should maintain a copy of the CPU/IPU scratchpad image in memory starting at memory location 300. The scratchpad memory image is used by both CPU and IPU during the power down sequence to roll out the CPU and IPU status and configuration words (table 2-6). During the power-up sequence, if memory has been preserved by battery backup, the CPU and IPU use the scratchpad memory image to auto restart.

Figure 2-8 illustrates the memory allocation for the scratchpad memory image, while table 2-7 lists the exceptions to the image that occur during power-down and power-up sequences.

The scratchpad memory image does not provide a unique location for the IPU trap table base address. This scratchpad location in the IPU is always defaulted to 0020 and, for all practicable purposes, is not relocatable.

2.9.2 Scratchpad Keyword

The scratchpad keyword is used to indicate the validity of the scratchpad contents. When the keyword is present in the scratchpad, the processor assumes that the software has loaded the scratchpad and set all options and features to the desired configuration. As long as the keyword remains in the scratchpad, the software settings of the scratchpad, features, and options will be retained through subsequent system reset functions.

If the keyword is not present in the scratchpad, the processor establishes the default setting for the table base addresses and all features and options.

The keyword is always forced to zero by the processor when the processor is initial program loaded or when the processor changes modes (CPU to IPU or IPU to CPU).

Diagnostic software that is testing the cache, ACS, WCS, or shared memory must always run with the keyword set to zero. The absence of the keyword ensures that the processor can re-establish the correct default settings for the processor options and features if a system reset occurs.

The scratchpad keyword in the scratchpad memory image is used to indicate the validity of the scratchpad memory image during power-up sequences with battery back-up. If the keyword is missing in the memory image, the processor assumes that the entire memory image is faulty and will not attempt to auto restart. Instead, the processor either auto-IPLs or auto-trap halts.

If the keyword is in the scratchpad image, the processor will attempt an auto restart, assuming the other conditions for auto restart have been met.

2.9.3 Device Entry

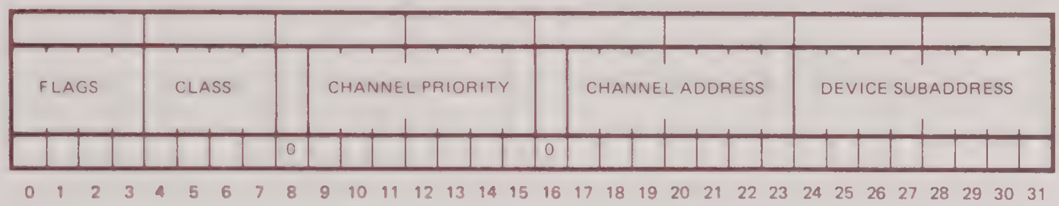
Figure 2-9 shows the format of the I/O device entry. The device entry associates the I/O protocol class with the channel interrupt priority level, the channel physical address, and the device subaddress. Device entries are stored in scratchpad locations 00 through 7F, according to the software channel address field of the entry.

2.9.4 Interrupt Entry

Figure 2-10 shows the format for the interrupt entry. The interrupt entry associates the I/O protocol class with the IOP/RTOM interrupt or the I/O interrupt identity, the interrupt priority level, the channel address, and the device subaddress. Interrupt entries are stored in scratchpad addresses 80 through EF. The software interrupt level, plus 80, provides an index to interrupt entries.

2.10 Control of Features and Options

The CPU and IPU provide features and options which are controlled and enabled/disabled by software. The


FLAGS
CLASS
CHANNEL PRIORITY
CHANNEL ADDRESS
DEVICE SUBADDRESS
0
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

FLAGS (UNARY, BITS 0-3)

BIT 0 = RAM LOADED (I/O CONTROLLER)

BIT 1 = PROGRAM VIOLATION (CLASS 0, 1, 2, AND E)

BIT 2 = ENABLE CHANNEL WCS EXECUTED (CLASS F)

BIT 3 = NOT USED

CLASS (BINARY, BITS 4-7)

VALUE	
0	CLASS 0 'TLC' LINE PRINTER (INVALID FOR 32/67 CPU/IPU)
1	CLASS 1 'TLC' CARD READER (INVALID FOR 32/67 CPU/IPU)
2	CLASS 2 'TLC' TELETYPEWRITER (INVALID FOR 32/67 CPU/IPU)
3	CLASS 3 IOP/RTOM INTERVAL TIMER AND FAST MULTIPLEXER SYSTEM (CPU ONLY)
7	EXTENDED I/O (CLASS F) IPU-IOP (IPU ONLY)
B	IPU-IOP INTERVAL TIMER (CLASS 3, IPU ONLY)
E	CLASS E STANDARD
F	CLASS F EXTENDED I/O

CHANNEL PRIORITY LEVEL (BINARY, BITS 8-15)

BIT 8	ALWAYS ZERO
BIT 9-15	SERVICE INTERRUPT PRIORITY LEVEL (ONES COMPLEMENT)

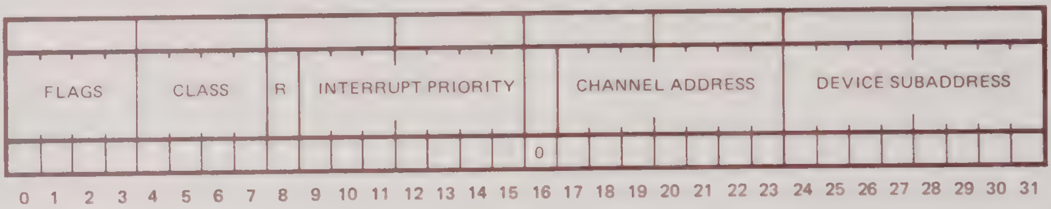
CHANNEL ADDRESS (BINARY, BITS 16-23)

BIT 16	ALWAYS ZERO
BITS 17-23	IOM, RPU, AND IOP CHANNEL PHYSICAL ADDRESS. (NOTE: IPU-IOP PHYSICAL ADDRESS MUST BE DIVISIBLE BY 4.)

DEVICE SUBADDRESS (BINARY, BITS 24-31)
(CLASS F AND CLASS 7 SUBADDRESS MUST BE ZERO)

830420

Figure 2-9. CPU/IPU Device Entry Format


FLAGS
CLASS
R
INTERRUPT PRIORITY
CHANNEL ADDRESS
DEVICE SUBADDRESS
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

FLAGS (UNARY, BITS 0-3)

BIT 0 = RAM LOADED (I/O CONTROLLER)

BIT 1 = NOT USED

BIT 2 = INTERRUPT ACTIVE

BIT 3 = INTERRUPT ENABLED

CLASS (BINARY, BITS 4-7)

VALUE	
0	CLASS 0 'TLC' LINE PRINTER (INVALID FOR 32/67 CPU/IPU)
1	CLASS 1 'TLC' CARD READER (INVALID FOR 32/67 CPU/IPU)
2	CLASS 2 'TLC' TELETYPEWRITER (INVALID FOR 32/67 CPU/IPU)
3	CLASS 3 IOP/RTOM INTERVAL TIMER AND FAST MULTIPLEXER SYSTEM (CPU ONLY)
6	IPU-IOP NON I/O INTERRUPTS (IPU ONLY)
7	EXTENDED I/O (CLASS F) IPU-IOP (IPU ONLY)
B	IPU-IOP INTERVAL TIMER INTERRUPT (CLASS 3, IPU ONLY)
E	CLASS E STANDARD CD-TD I/O
F	CLASS F EXTENDED I/O (CPU ONLY)

R (BIT 8) = 1 IOP/RTOM INTERRUPT (NON I/O) (CPU/IPU)
= 0 I/O INTERRUPT (CPU/IPU)

INTERRUPT PRIORITY LEVEL (BINARY, BITS 9-15)
SERVICE INTERRUPT PRIORITY LEVEL (ONES COMPLEMENT)

CHANNEL ADDRESS (BINARY, BITS 16-23)

BIT 16	ALWAYS ZERO
BITS 17-23	IOM, RPU, AND IOP CHANNEL PHYSICAL ADDRESS. (NOTE: IPU-IOP PHYSICAL ADDRESS MUST BE DIVISIBLE BY 4.)

DEVICE SUBADDRESS
(CLASS F AND CLASS 7 SUBADDRESS MUST BE ZERO)

830419

Figure 2-10. CPU/IPU Interrupt Entry Format

following text describes their operational characteristics and the software techniques required to control them.

2.10.1 PROM or ACS Modes

The programmable read only memory (PROM) mode and the alterable control store (ACS) mode are two mutually exclusive modes of operation for the 32/67 processors. They describe the source of firmware for the processor, either PROM or ACS. The PROM or the ACS mode is enabled or disabled by the SETCPU instruction. The operating mode can be monitored by the software read status (RDSTS) instruction.

2.10.1.1 PROM Mode

The PROM mode is the standard source of the processor firmware. The PROM Mode is entered at each powerup, system reset, and IPL sequence. This mode remains in effect until the processor is forced into the ACS mode by software and the SETCPU instruction.

Once the system has been operating in the ACS mode, and a system reset occurs, the processor returns to the ACS mode during the associated firmware reset sequence only if the scratchpad keyword is correct. If the keyword is incorrect, the processor remains in the PROM mode until a software command is issued.

Firmware contained in the PROM array can only be altered by returning the processor to the manufacturer for PROM integrated circuit replacement.

The contents of the PROM may be read under software control by means of the read writable control store (RWCS) instruction.

2.10.1.2 ACS Mode

The ACS is the alternative source of processor firmware. It consists of a random access memory (RAM) array containing 4096 locations of 64-bit microwords. Writes to ACS and reads from ACS are accomplished by software.

The purpose of the ACS is to provide a method by which software can be used to update or enhance the firmware without having to return the processor to the manufacturer for PROM replacement.

The normal sequence for using the ACS is as follows.

1. At powerup, the firmware reset sequence causes the processor to operate in the PROM mode.
2. During the software initialization sequence, software loads the ACS RAMs.
3. Software verifies (reads and compares) the contents of the ACS RAMs.
4. Software causes the processor to enter the ACS mode of operation.

The write to writable control store (WWCS) instruction must be used by the software to write to the ACS RAMs, with an ACS address in the range 0000 to 0FFF hex. In order for the WCS instruction to acutally modify the contents of the ACS RAMs, the following software-controlled processor conditions must be met.

1. The processor must be operating in the PROM mode.
2. The enable write to WCS bit must be set in the processor status word. This control bit may be set or reset by the SETCPU instruction and monitored by the RDSTS instructions.

In order to read from the ACS RAMs, the software must use the read writable control store (RWCS) instruction, with an address in the range 0000 to 0FFFF hex. The PROM and ACS addresses have the same range and appear identical. Therefore, bit 0 of the 32-bit PROM/ACS address is used to differentiate between read ACS versions and read PROM versions of the RWCS instruction. If bit 0=1, ACS is read. If bit 0=0, PROM is read.

When the ACS load function is complete, software should clear the enable WCS write bit in the processor status word. This prevents inadvertant WWCS instructions from destroying the contents of the ACS or the WCS RAM arrays.

Typically, the ACS may be loaded with patches or with an entirely new firmware image. Usually, when the ACS is patched, the software initially copies the contents of the PROMs into the ACS and then changes the ACS locations to be patched to the new firmware. When entirely new firmware is loaded into the ACS, the software ususally clears all of the ACS RAMs initially then loads the new firmware into the ACS.

By software and firmware convention, locations 0FFD, 0FFE, and 0FFF of the PROM and ACS are reserved locations and do not contain standard microwords. These reserved locations are used as follows:

1. Location 0FFD contains the revision control number of the firmware set. The revision control number includes the development source and object file name for the firmware set, plus a subset of the 12-digit Gould S.E.L. part number of the firmware set.

2. Location 0FFE contains the 64-bit checksum of the firmware set. The checksum is computed by generating the 64-bit sum of all microwords between 0000 and 0FFD, and then one's complementing the 64-bit sum and storing it in location 0FFE.

3. Location 0FFF is reserved by the development support system/diganostic processor, and the content of this location is unpredictable. This location is excluded from checksum calculations.

If the processor is operating in the ACS mode, it reverts to the PROM mode at each system reset. If the scratchpad keyword is correct, the processor firmware reset sequence causes the processor to return to the ACS mode. The ACS contents are not altered by system reset.

During power-up or IPL sequences, the processor always defaults to the PROM mode of operation. It remains in the PROM mode until it is commanded to the ACS mode by software. If the software commands the processor to the ACS mode, and the ACS has not been loaded, the results are unpredictable. Therefore, the ACS should always be loaded as soon as is feasible in the software initialization sequence.

2.10.1.3 ACS Mode Special Considerations

The following considerations must be taken into account when firmware is being developed for ACS applications.

1. All decode vectors for macroinstructions reside at dedicated locations. Similarly, microinterrupt vectors and decode exception vectors reside at dedicated locations. ACS applications cannot alter the vector address; each possible vector must be accounted for as in the initial PROM firmware set.

2. When the processor changes from the PROM to ACS or ACS to PROM mode, it is effectively executing a firmware branch. Firmware in the PROM and the ACS should be identical where these changes occur. These mode changes can only occur in the system reset flow and in the SETCPU instruction flow.

2.10.2 Development Support System/Diagnostic Processor Mode

NOTE

Support of this mode of operation is not available at initial product release. The following information is provided to supplement the PROM and ACS mode text.

This is a third optional mode of operation for the 32/67 processors. The functionality for this option is designed into the processors. Comprehensive software, designed for PROM and ACS support, should be cognizant of the effect of this optional mode on the PROM and the ACS modes of operation.

This option provides snapshot and monitoring capability of the processor microoperation and the capability of replacing the processor PROM control store with a read only memory simulator (ROMSIM) function. The snapshot and monitor functions have no effect on the operational characteristics of the processor. However, the ROMSIM function prohibits software from altering the contents of the ACS.

The operator commands of this option allow it to operate in either of two control store modes, the PROM mode or the ROMSIM Mode. The PROM mode allows the processor to operate as previously described and the presence of the option is transparent to the processor software. The ROMSIM mode uses the ACS RAMs to hold the processor control store firmware, and the software cannot write to the ACS. Also, software cannot command the processor to operate in the PROM mode. However, the software can read the ACS (ROMSIM) or the PROMS.

When the option is operating in the PROM mode, processor software cannot detect the presence of the option. Operation in the ROMSIM mode is indicated by the ROMSIM status bit in the processor configuration word. Software can read the processor (CPU/IPU) configuration word by executing a read configuration word variation of the read program status word two (RPSWT) instruction.

2.10.3 Cache

For the purpose of a description biased toward the software operation of the cache, the cache can be considered as four independent banks. Each cache bank is 2048 locations deep and 32 bits wide. There are four index arrays, one associated with each cache bank. An index array contains the upper 11 bits of the associated memory address plus a hit (valid) bit. Two of the four cache banks are reserved instructions, the other two banks are reserved for operands. Therefore, instructions cannot be fetched from the operand cache and operands cannot be fetched from the instruction cache.

The cache control logic is always prefetching to determine if the the required data (instruction or operand) is cache resident. If the data is not in the cache (cache miss), the control logic fetches the data from memory and stores it in the cache (assuming the cache is enabled). The address source for instruction prefetching is the software macroprogram counter. The address source for operand prefetching is the logical memory address register. The cache control logic never prefetches operands unless commanded by the processor IE Unit. When prefetching or operand stores are not in progress (commanded by the IE Unit), the cache prefetches instructions. These instructions are presented to the instruction pipeline in the IE unit, which may accept or reject the prefetch. If the prefetch is accepted, the macroprogram counter address is advanced and the next logical program counter address is prefetched. If the prefetch is rejected, the same logical program counter address is refetched. This process is repeated until the prefetch is accepted by the IE Unit pipeline. The macroprogram counter (instruction address) and the logical memory address register (operand address) both provide logical addresses. Therefore, the cache control logic must convert the logical address to a physical address on each cache transaction. The map logic in the CS Unit performs the logical-to-physical address conversion.

The minimum data quantity fetched from memory by the cache control logic is one word (32 bits). However, prefetching is normally done on a doubleword basis. Doubleword prefetching is accomplished by two successive single-word memory reads for the even and odd addresses of a doubleword pair. The even and odd pair may be prefetched in any order (even-odd or odd-even) depending upon the specific word address contained in the macro-program counter or the logical memory address register. If the software design contains intermixed instructions and operands, doubleword prefetching can intermix instructions and operands in the operand cache or the instruction cache. This intermixing produces false cache misses because operands cannot be fetched from the instruction cache, and instructions cannot be fetched from the operand cache.

For store type instructions, the cache control logical may store bytes, halfwords, or words to cache and memory as required. Doubleword stores are accomplished by two successive single word stores. For single and doubleword stores, both cache and memory are written to. For byte or halfword stores, memory is always written to, but cache is only written to if a cache hit occurs (addressed word is in the cache). For byte or halfword cache writes, only the specified byte or halfword of the 32-bit word is changed; the remainder of the 32-bit word is unchanged. During cache writes caused by store instructions, only the operand cache is modified. During store instructions, if an instruction cache hit is detected by a cache write, the instruction cache entry is invalidated.

For an externally initiated memory write, the cache is read on the next SelBUS cycle. If a cache hit is detected at the address specified by the external write, the cache location is invalidated.

Doubleword prefetching can cause operands to be in the instruction cache and instructions to be in the operand cache. Therefore, if an instruction or an operand cache read detects a hit in the wrong cache and a miss in the correct cache, a cache miss occurs. The hit data is discarded, the location is invalidated, and the cache control logic fetches the specified data from memory. If a cache read detects a cache hit in both instruction and operand caches, the incorrect type cache hit is invalidated, the data is discarded, and the data from the correct type hit (operand or instruction) is used. This prevents a subsequent memory fetch.

When SelBUS memory read data return transfers are loaded into the cache, they are loaded by read transfer type (operand or instruction). The opposite cache is not checked for a valid bit (cache hit) at the specified address. Therefore, a specific data element (operand or instruction) may reside in either or both of the caches. Dual residency of a data element is always corrected the next time the data element is referenced by software.

When an IE Unit (firmware generated) cache read results in a cache miss, the cache control logic fetches a single word from memory instead of the standard doubleword memory fetch.

2.10.4 Cache Control

The cache can be enabled, disabled, and cleared by software using the cache memory control (CMC) instruction. The default state of the processor cache is all banks turned on. The disable cache functions are provided for diagnostic software cache fault isolation. The operational state (enabled/disabled) of the cache banks are monitored by the processor configuration word and the software read configuration word varitation of the RPSWT instruction.

The cache is also indirectly controlled by the shared memory control (SMC) instruction and the IOP panel address stop functions.

The following subordinate paragraphs describe the various control functions in detail. Detailed information on specific bit formats for the CMC and SMC instructions are contained in the 32/67 Reference Manual.

2.10.4.1 CMC Instruction

In general, the CMC instruction provides four bits which enable or disable the corresponding four banks of the cache (two operand banks and two instruction banks). Four bits are provided to initialize the four cache banks, and one bit is used to enable/disable the instruction cache during instruction prefetching.

The four enable/disable bits cause a cache bank to be enabled when the corresponding bit is set to 1; a bank is disabled when the corresponding bit is set to 0. A disabled cache bank does not track (copy) system memory activity and its contents cannot be changed. If a disabled cache bank is enabled, its contents are the same as at the time it was disabled. Therefore, a cache bank should never be enabled without first initializing it. The CMC instruction that enables a cache bank may also initialize it.

The CMC instruction initialize bits cause all of the valid bits in the corresponding cache bank index array to be set to zero. This effectively clears the cache bank.

If a cache miss occurs due to a disabled cache bank, the control logic fetches a doubleword pair from memory. However, the data returned from memory is not copied into the cache.

The cache enable/disable instruction prefetch bit causes instruction read misses when it is set ot zero. It enables instruction read hits when it is set to one. When this bit is set to one, the cache operates normally. When set to zero, all instruction reads result in cache miss; however, all system memory activity is copied into the instruction cache. If a cache miss occurs due to a disabled instruction prefetch, a doubleword is fetched from memory and copied into the instruction cache. Software may enable an instruction prefetch without initializing the two instruction cache banks (assuming that the instruction cache banks are enabled).

2.10.4.2 SMC Instruction

The following description is restricted to the effects of the SMC instruction on the cache. The SMC instruction may enable the read and lock mode, or it may describe the upper and lower boundaries of a shared memory region.

When the read and lock mode is enabled, any read and lock cache transaction issued by the IE Unit (caused by SBM or ZBM instructions) causes the cache logic to force a cache miss and to issue a single read and lock transaction to the SelBUS and memory. The data returned from memory is copied into the cache and transferred to the IE Unit. Subsequently, when the IE Unit issues a write and unlock, the data is written into the cache and sent to the SelBUS and memory together with a write and unlock transaction code. Since the write and unlock data is written to the cache, any subsequent non read and lock access of the memory location (i.e., TBM instruction) should produce a cache hit.

When the SMC instruction is used to define shared memory boundaries, there are two major effects on the cache:

1. Any cache transaction within shared memory boundaries automatically produces a cache miss, and all associated data is not copied into the cache.

2. Any cache read within shared memory boundaries results in a singleword fetch instead of a doubleword fetch. This reduces SelBUS and shared memory traffic.

2.10.4.3 IOP Panel Address Stop

The IOP panel address stop functions are fully described later in this chapter; the following description is restricted to their effects on the cache.

The panel read stop function includes operand read stop (RS), instruction read stop (IS), and address stop (AS). When a read stop is in effect, the CPU and IPU caches are disabled and all cache read instructions produce cache misses. Singleword fetches are used instead of doubleword fetches to reduce SelBUS traffic. The associated data is copied into the cache.

Both CPU and IPU caches are disabled in order to keep processor execution speeds symmetrical. The IOP panel write stop function does not disable the cache.

2.10.4.4 Instruction Store (nonpresent memory)

Store instructions that produce nonpresent memory cause the processor firmware to initialize (clear) all the cache on each occurrance. The clear cache function is required because all stores (cache writes) occur before SelBUS nonpresent memory is detected.

2.10.4.5 Cache System Reset

The firmware system reset flow, or any sequence that causes a firmware reset (auto restart, auto IPL, or IPL), cause all cache banks to be initated (cleared). If the scratchpad keyword is correct in the processor scratchpad during the firmware reset, the CMC and SMC control functions revert to the state defined in the processor configuration word. If the scratchpad keyword is incorrect, the CMC and SMC cache control functions assume their default states (i.e., all cache banks enabled and SMC functions disabled).

2.10.5 Cache Memory Prefetch

The following text provides a summary of the cache memory prefetch rules.

Data is always fetched from memory in doublewords (i.e., two back-to-back singleword SelBUS fetches) unless the specific transaction is covered by one or more of the following rules.

1. The IE Unit (firmware) issues the cache read transaction.
2. The cache is disabled by the IOP panel address stop functions.
3. The SMC read and lock is enabled and the IE Unit (firmware) issues a read and lock transaction.
4. The SMC shared memory boundaries are enabled and the effective physical address of the cache read is within shared memory boundaries.

In the preceding rules, cache misses are forced and singleword reads are issued to the SelBUS memory unless the specific software instruction requires a doubleword. In this case, the cache control logic issues two singleword SelBUS reads for the doubleword pair.

2.10.6 Shared Memory

The shared memory features include allowing individual processors of a multiprocessor system to communicate through a common block of memory. Specifically, these are the read and lock feature and the shared memory boundary feature.

Two types of shared memory configurations are common with 32/67 multiprocessor systems. One configuration is a CPU and IPU system, in which the multiprocessors and shared memory reside on a common bus as depicted in figure 2-11. In this configuration, each processor can monitor SelBUS memory write activity and keep its cache up to date. The only

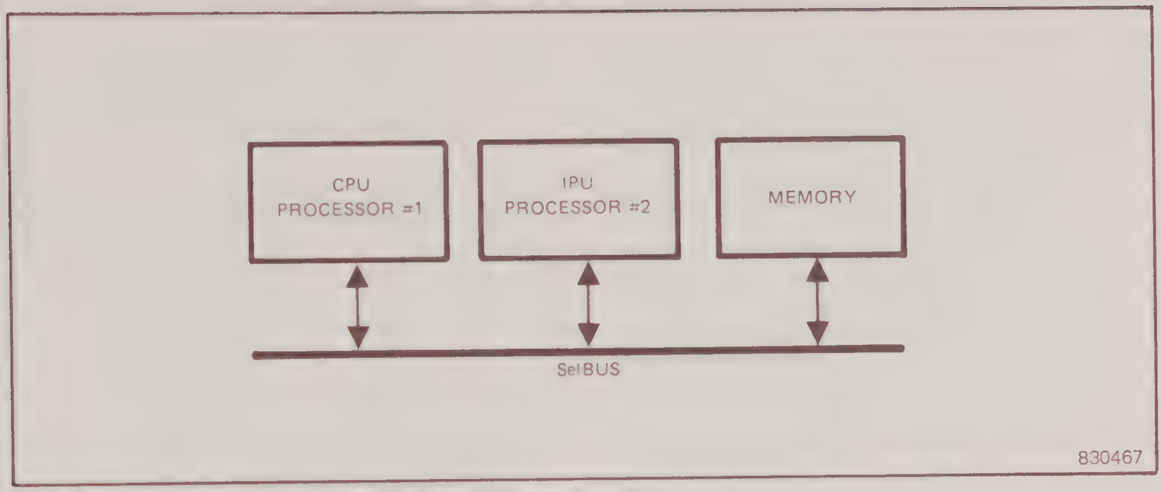


Figure 2-11. CPU/IPU Shared Memory

shared memory feature that is required for a CPU/IPU shared memory is the read and lock feature. This is required to implement interprocessor bit semaphores described later.

The second type of shared memory consists of a multiple (two or more) SelBUS configuration, and a processor(s) associated with each bus. Each SelBUS communicates to a common multiported memory to form a multiprocessor (SelBUS) system. This configuration is illustrated in figure 2-12. In this configuration, the processor(s) on SelBUS 1 have no knowledge of the memory activity on SelBUS 2. Therefore, the cache associated with the processor(s) on SelBUS 1 cannot be kept up to date with respect to the memory write activity generated over SelBUS 2. The reverse is also true for processor(s) cache associated with SelBUS 2 for write activity generated over SelBUS 1. In this configuration the caches are not used when memory acceses are made within the shared memory region. To implement this requirement, each of the processors has shared memory boundary registers which describe the upper and lower boundaries of the shared memory region. Any memory access within these boundaries forces a cache miss, causing the processor to access the multiported shared memory. In the remote shared memory configurations, the read and lock features must be used to implement interprocessor bit semaphores.

2.10.6.1 Interprocessor Bit Semaphores

Interprocessor semaphores are software defined bit flags. These may be used to enable or disable access of a processor to a defined region within shared memory. Frequently, a bit flag is defined to indicate when one processor is modifying a memory region. Access to that region is prohibited to other processors until memory modification is complete and the bit flag is cleared by the first processor.

Since bit flags may be used to control processor access to memory, multiple processors may attempt to test or modify flags within a flag word simultaneously. With a standard (non read and lock) implementation of the bit in memory instructions, confusion can result between processors, associated caches, and memory. The read and lock feature can resolve the bit confusion. The following text describes how the bit confusion occurs in a non read and lock implementation, and how the read and lock eliminates this confusion. A description is also provided of the processor internal implementation of the bit in memory instructions which include test bit (TBM), zero bit (ZBM), and

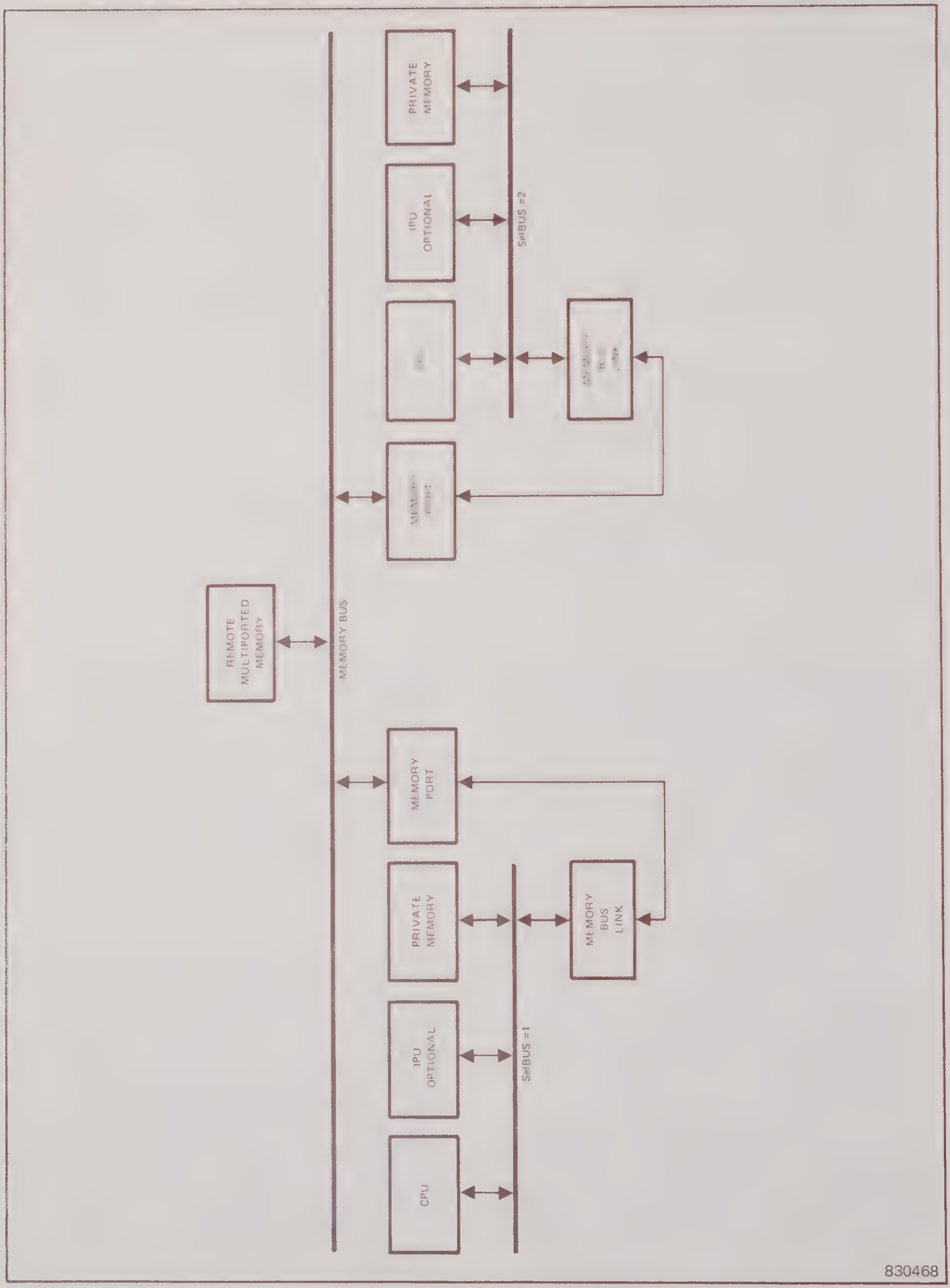

PRIVATE MEMORY
IPU OPTIONAL
SelBUS #2
MEMORY BUS
REMOTE MULTIPORTED MEMORY
MEMORY PORT
PRIVATE MEMORY
MEMORY BUS LINK
IPU OPTIONAL
SelBUS #1
CPU
830468

Figure 2-12. Remote Shared Memory Configuration

set bit (SMB) instruction. Add bit (ABM) is deliberately excluded from the description since ABM should never be used for semaphore manipulation.

In a non read and lock environment, the TBM, ZBM, and SBM instructions are implemented by the following sequence.

1. The processor reads the 32-bit word containing the specified bit from memory.
2. The processor sets condition codes based upon the initial state of the specified bit. At this point, the TBM is complete.
3. For the ZBM and SBM instructions, the processor modifies the bit in the 32-bit word.
4. The processor writes the updated 32-bit word to memory.

The entire sequence for ZBM and SBM requires a minimum of seven SelBUS cycles. This allows a number of available SelBUS (memory) cycles for second processor to access the same bit and/or word as the first processor. If the second processor is also executing a SMB or ZMB instruction, which requires a read-modify-write sequence, the first processor would write its modified data word to the SelBUS and memory followed by the second processor. The SelBUS and memory word write by the second processor overlays the data written by the first processor. This causes any semaphore modification generated by the first processor to be lost and undetected by the second processor. Table 2-10 shows a multiprocessror non read and lock SelBUS sequence example.

Use of the read and lock feature with the SBM and ZBM bit in memory instructions prevents a second processor from inadvertantly destroying bit flags generated by a first processor. This assumes that both processors use only SBM and ZBM instructions to modify bit flags. The read and lock feature is not applicable to the TBM or ABM instructions.

The read and lock feature functions by replacing the normal SelBUS read transaction of the SBM and ZBM with a read and lock SelBUS transaction. This causes the memory module to reject any subsequent non-write and unlock transaction to the same address with memory busy status. The normal word write of the SBM and ZBM sequence is replaced with a SelBUS write and unlock transaction, which causes the memory module to resume traffic to the address specified in the read and lock and write and unlock transactions. The memory address of the two transactions must always be identical.

Different types of memory modules (different model numbers) react differently to read and lock sequences. However, the end result of the read and lock is to prevent inadvertant flag destruction of concurrent SBM and ZBM sequences by multiprocessors.

The integrated memory module (IMM), which is standard for 32/67 systems, must be jumpered to enable the read and lock mode (full details are contained in the 32/67 System Manual). When the read and lock is received, the specified address and all addresses within the IMM range become locked. This prevents all memory traffic to the IMM until the write and unlock transaction is received. The side effect of locking the entire range (which may be one megabyte) is to reduce system memory bandwith significantly. The reduction in memory bandwidth can be amplified greatly by cache based processors if software uses SBM or ZBM instructions in tight program loops with read and lock enabled. Example:

```
SBM  Bit01, FLAGS
BCT  SET, $ -1W
```

The processor would remain in this loop until an outside function (second processor) clears flag bit 01. If the SBM and BCT instructions are cache resident, only the read and lock and the write and unlock transactions would go to the SelBUS and memory. With the 32/67 processor, the memory containing the

Table 2-10
Multiprocessor - Non Read & Lock SBM, Example

Location X'1000' = 0000 0000
Processor #1 SBM Bit 01, X'1000'; Processor #2 SBM Bit 02, X'1000'

Cycle	SelBUS	By or to Processor #	Processor #1 Internal	Processor #2 Internal
1	Word Read	P2		
2	Word Read	P2		
3	Empty			
4	Word Return =00000000	P1		
5	Empty		Load Cache =00000000	
6	Word Return =00000000	P2	Set bit 01	
7	Word Write =40000000	P1	Load Cache =4000000	Load Cache =00000000
8	Empty			Invalidate Cache & Set Bit 02
9	Word Write =20000000	P2		Load Cache =20000000
10			Invalidate Cache	

Final Contents:
Location 1000 = 20000000
Cache P1 = invalid
Cache P2 = 20000000
Memory write by P1 is lost

word flags would be locked for 50 to 60 percent of the time. This would have a significant effect on system available memory bandwith. A preferred method of providing the same functionality is as follows:

```
   TEST  TBM  Bit01, FLAGS
         BCT  SET, $ -1W
         SBM  Bit01, FLAGS
         BCT  SET, TEST
 CONTINUE EQU  $
```

In this sequence, if bit 01 is initially set, the processor holds in a loop that does not use read and lock until bit 01 is cleared by a second processor.

2.10.6.2 Interprocessor Semaphore Considerations

For an interprocessor semaphore scheme to be effective, the following rules and suggestions should be observed.

1. All processors in a multiprocessor system must use the read and lock feature.

2. A memory word containing semaphores (bit variables) must not contain other types of variables (byte, halfword etc.). Semaphore words may contain constants providing the constant is not modified.

3. Only the SBM and ZBM instructions should be used to modify memory words containing a semaphore.

4. Software must avoid the use of tight read and lock loops. If a holding loop on a semaphore is required, the TBM instruction should be used to detect the semaphore change of state.

Table 2-11 contains a sequence where one processor is using a TBM to test a flag, and a second processor uses a zero memory word (ZMW) to clear the flags. The ZMW is a non read and lock instruction. Therefore, the cache of the first processor becomes unsynchronized with the corresponding memory location.

2.10.6.3 SMC Read and Lock Control

The following paragraphs describe the control of the read and lock feature by the shared memory control (SMC) instruction.

The read and lock feature is enabled or disabled by bit 07 of the register specified in the SMC instruction. IF bit 07=1, read and lock is enabled across the entire address range of the processor. The read and lock feature operates only with the SBM and ZBM instructions. It does not operate with the TBM or ABM instructions. The read and lock feature causes the cache control logic to force a cache miss on IE Unit generated (firmware) read and lock transaction codes. A read and lock singleword read is issued to the SelBUS and memory. The write and unlock transaction code can only be issued by the IE unit (firmware). During the write and unlock sequence, the data associated with the write is loaded onto the cache.

During SBM and ZBM instruction sequences, the I Unit checks the cache and map for potential errors prior to enabling the E Unit read/lock and write/unlock sequences. If errors are detected, the execution phase of the SBM or ZBM is aborted and a trap is initiated. The precheck of errors includes map invalid, map write protection (privilege violation), non present memory, and parity error (uncorrectable data error).

The read processor configuration word variation of the read processor status word two (RPSWT) can monitor the state of the enable/disable read and lock bit. When changing the state of this bit, software should always precede the SMC instruction with a read processor configuration word, merge the new state of the read and lock bit with the configuration word, and then execute the SMC instruction.

Read and lock must be enabled separately in the CPU and IPU. The read and lock mode must be used in a CPU and IPU configuration.

2.10.6.4 SMC Shared Range Control

The following paragraphs describe the control of the shared memory ranges provided by the SMC instruction, and the direct effects on the cache/SelBUS logic.

Bit 01 of the register specified in the SMC instruction enables or disables the shared memory range feature (1=enabled, 0=disabled). When the enable is active, bits 02 through 06 of the register provide the high boundary and bits 08 through 12 provide the low boundary of the shared memory range. The two boundary fields provide the five most significant bits of the 24-bit memory physical address. This physical address divides the total memory address (four megawords) into 32 equal ranges of 128 kilowords each. The two

Table 2-11
Multiprocessor Use of Non Read and Lock Instruction to Modify Semaphore, Example

Processor #1 TBM Bit 01, X'1000'; Processor #2 ZMW X'1000'
BCT SET, $-1W

Cycle	SelBUS	By or To Processor #	Processor #1 Interval	Processor #2 Interval
1	Word Read	P1		
2	Empty			
3	Word Write =00000000	P2		Load Cache =00000000
4	Word Return =FFFFFFFF	P1	Invalidate Cache	
5	Empty		Load Cache =FFFFFFFF	
6	Empty		Test Bit 01 and branch back to TBM which produces cache hit. Note: Processor 1 continues in the loop until some external event causes cache to be invalidated	
Final results for location X'1000': Memory = 00000000 Cache Processor #1 = FFFFFFFF Cache Processor #2 = 00000000				

boundary words may specify any single 128KW range, any contiguous group of 128KW ranges, or the entire 4MW (16 megabyte) address range as the shared memory range.

To define the first 128KW of memory as shared, the low and high boundaries are defiend as zero. This designates memory locations 000000 through 07FFFC hex as shared.

To define the first two 128KW groups of memory as shared, the lower limit is defined as zero and the upper limit is defined as one. This designates memory locations 000000 through 0FFFFC hex as shared.

In all cases, the low and the high physical address boundaries for shared memory must contain the five most significant bits of the 24-bit boundary address. If the low and high boundaries are equal, a single

128K range is shared. If the high boundary is 1F and the low boundary is 00 hex, the entire 4MW address range is shared.

The enabled state of the shared memory range logic automatically enables the read and lock mode to any SBM and ZBM instruction which has an effective physical address within the shared range. This function overrides the control normally provided by the enable/disable read and lock bit of the SMC instruction.

All instruction fetches or operand fetches, that have a physical address within the shared range, cause a cache miss and a SelBUS singleword read from memory. The data returned from memory is not stored in the cache. For all instructions which cause memory writes (if the write address is within shared boundaries), the associated write data is not stored in the cache. By reverting to singleword prefetching in the shared memory range, the processor reduces superfluous SelBUS and memory activity. By not storing the data associated with the shared memory range in the cache, the processor eliminates cache contention between operands/procedures which may be stored in the cache and operands/procedures which cannot be stored in the cache.

The read processor configuration word variation of the read processor status word two (RPSWT) instruction monitors the state of the enable shared memory bit and the high and low boundary registers. When software is changing the state of the shared logic, the SMC instruction should always be preceded by a read processor configuration word, a merger of the new state of the shared memory with the configuration word, and then the SMC instruction is executed.

Shared memory ranges must be enabled separately in the CPU and IPU processors.

2.10.7 Writable Control Store (WCS)

The following paragraphs describe software initialization and maintenance of the WCS feature.

The 32/67 processor provides 8K by 64-bit WCS locations as a standard feature. For descriptive purposes, the WCS may be considered as add on control store to the PROM or ACS control store. The PROM and ACS control store is provided for the standard instruction set. The WCS control store is provided for a user supplied instruction set or firmware applications.

Internally, the WCS is divided into two 4K WCS banks, which are configured by firmware and initialized separately. The two 4K banks correspond to the physical organization of the integrated circuits residing on the MS Unit. At processor initialization (which includes auto restart, auto IPL, IPL and system reset), with an incorrect scratchpad keyword, the firmware writes a 'long branch to undefined instruction' microword into each location in the two WCS banks. The firmware then reads each location zero relative to the WCS banks. If the write and read data is equal, the entire 4K bank of WCS is considered present and operable, and it is marked present in the processor configuration word. If the write data and read data are not equal, the entire 4K bank is marked non present in the processor configuration word.

In subsequent system reset sequences, if the scratchpad keyword is correct, the firmware does not destroy the contents of the WCS. It maintains the WCS and the WCS configuration in the configuration word through the system reset sequence.

Software initialization of the WCS should occur during software system initialization. The software has two instructions available to initialize and maintain WCS: read WCS (RWCS) and write WCS (WWCS). Any software read or write sequence should always be preceded by a read processor configuration word variation of the read processor status word two (RPSWT) instruction. The processor configuration word indicates the firmware status of the two 4K WCS banks (i.e., present or non present/inoperable). If software attempts a RWCS or a WWCS instruction, and the WCS target address resides in a WCS bank that is non present or inoperable, an address specification trap occurs. Firmware defines a WCS bank as non present or inoperable if location zero (relative to the 4K bank) cannot be written to, read from, or compared. Whenever an indication is provided to the software that a WCS bank exists, but the configuration word indicates that the bank is non present or inoperable, the software should notify the system operator through a console error message. The control store diagnostic program can then be used for fault isolation.

The RWCS instruction is an unprivileged halfword instruction that specifies two general purpose registers. The register specified by the RS field of the instruction provides the WCS target address and must be in the range 1000 through 2FFF hex. If the target address is less than 1000, the RWCS instruction is specifying a PROM or ACS location. PROM or ACS depends upon the state of RS bit 00 which, when set to 1, specifies an ACS read. If the WCS target address is greater than 2FFF, or if the address resides in a 4K WCS bank that is marked non present or inoperable, an address specification trap occurs. The RD field of the RWCS instruction specifies the doubleword memory address that receives the 64-bit content of the specified WCS location. If the memory address is not doubleword bound, an address specification trap occurs.

The WWCS instruction is a privileged halfword instruction that specifies two general purpose reigsters. The register specified by the RD field of the instruction provides the WCS target address and must be in the range 1000 through 2FFF. If the target address is less than 1000, the WWCS instruction is specifying an ACS location. If the WCS target address is greater than 2FFF, or if the address resides in a 4K WCS bank that is marked non present or inoperable, an address specification trap occurs. The RS field of the WWCS instruction specifies the doubleword memory address of the 64-bit microword that is written to the specified WCS location. If the memory address is not doubleword bound, an address specification trap occurs.

Write to WCS sequences must always be preceded by a SETCPU instruction. This instruction enables WCS/ACS writes (sets CPU status word bit 20, bit 20 at 1 = enable write). If a WWCS instruction is excuted and WCS writes are not enabled, an undefined instruction trap occurs. When the write WCS sequence is complete, writes should be disabled by a SETCPU instruction that clears bit 20 of the CPU status word. The SETCPU instruction must always be part of read status word (RDSTS), modify bit, and SETCPU instruction sequence as previously described.

After the WCS has been loaded, the software should always verify (read and compare) the WCS contents to ensure that the WCS RAMs are operable and will retain the WCS microcode.

User software utilizes the jump to WCS (JWCS) instruction to actually enter and initiate execution of WCS microcode. The JWCS instruction is an unprivileged fullword instuction with a memory reference format. The instruction effective byte address is used as the WCS entry point address. The final effective address must be in the range 1000 through 2FFF or an address specification trap occurs. If the effective address specifies a 4K WCS bank that is marked non present or inoperable, an address specification trap occurs If the effective address specifies a legal WCS address that has not been written by software, an undefined instruction occurs. This is due to the fact that the entire WCS address range was initialized to branch to undefined instruction during the last firmware processor reset.

2.10.8 Floating-point Accelerator (FPA) Option

This two-board option cables to the CS Unit. It requires two SelBUS slots, but does not use the SelBUS except for power, ground, and clocks. For a complete description of the FPA, refer to the CONCEPT/32 Floating-point Accelerator Model 3611 Technical Manual, Publication Order Number 303-003020-000.

The FPA, when present and on line, may be enabled or disabled by software using the SETCPU instruction to modify CPU status word bit 22. When bit 22=1, the FPA is disabled. When bit 22=0, the FPA is enabled.

The enabled/disabled state of the FPA is monitored by the CPU status word bit 22 and can be software monitored by using the read status word (RDSTS) instruction. The bit definition for status word bit 22 is:

Bit 22=1	1. The FPA is non present OR 2. The FPA is off line OR 3. The FPA is software disabled.
Bit 22=0	The FPA is present, on line, and software enabled.

If the software clears bit 22, but it remains set in the status word, the FPA is unavailable to software as either non present or off line. The on line/off line state of the FPA is controlled by a switch located on the FPA board set.

When the FPA is present and on line, its state is software enabled. The software enabled state is established by the processor firmware at the end of the firmware system reset sequence. The contents of the scratchpad keyword does not effect the FPA state during the reset sequence. In actual usage, the hardware reset signal disables the FPA, but firmware always enables the FPA at the end of the subsequent firmware reset.

When the FPA is enabled, on line, and present, it is used by the processor firmware to improve the performance of the following instructions.

1. Add floating-point word (ADFW).
2. Subtract floating-point word (SUFW).
3. Multiply floating-point word (MPFW).
4. Divide floating-point word (DVFW).
5. Add register floating-point word (ADRFW).
6. Subtract register floating-point word (SURFW).
7. Multiply register floating-point word (MPRFW).
8. Divide register floating-point word (DVRFW).
9. Add floating-point doubleword (ADFD).
10. Subtract floating-point doubleword (SUFD).
11. Multiply floating-point doubleword (MPFD).
12. Divide floating-point doubleword (DVFD).
13. Add register floating-point doubleword (ADRFD).
14. Subtract register floating-point doubleword (SURFD).
15. Multiply register floating-point doubleword (MPRFD).
16. Divide register floating-point doubleword (DVRFD).
17. Float integer word to floating-point word (FLTW).
18. Float integer doubleword to floating-point doubleword (FLTD).
19. Multiply memory byte (MPMB) - fixed point.

20. Multiply memory halfword (MPMH) - fixed point.

21. Multiply memory word (MPMW) - fixed point.

22. Multiply register to register (MPR) - fixed point.

In CPU and IPU systems, the FPA option must be configured symmetrically on both systems.

It should be noted that the quotients produced by the firmware implementation of the divide floating-point instructions differ in accuracy from those quotients produced by the FPA option. This difference is due to the fact that the two floating point implementations use different divide algorithms.

2.11 Operator Indicators

The turnkey panel (figure 1-3) is equipped with LED indicators which denote the operational state of the CPU and of the IPU (if the system is configured with this second processor). A description and definition of the HALT, RUN, WAIT and INTRPT indications is provided in the following paragraphs. Additionally, the processor state indicators are displayed on the IOP console CRT whenever an IOP panel command is executed.

2.11.1 Halt Indicator

The true state of the HALT indicator denotes that the CPU/IPU has stopped processing software instructions and has stopped scheduling software interrupts and errors. Any interrupts or errors that become pending while the CPU/IPU is halted, will remain pending until the CPU/IPU enters the run state or a reset function is executed. If I/O was in progress when the CPU/IPU was halted, the I/O will continue to normal completion, except that the I/O termination interrupt will not be processed.

The CPU/IPU may be placed in the halt state by any of the following events:

1. The execution of a software halt instruction when the CPU/IPU is operating in a privileged state with the privileged state, halt trap disabled
2. The execution of a control panel, halt command
3. The occurrence of an address compare, stop signal from the control panel
4. The occurrence of an I/O or memory error during power-up auto-restart sequence or an IPL sequence (automatic trap halt)
5. The occurrence of a software error trap while software traps are disabled (automatic trap halt).

When the processor is in the halt state, the firmware is executing an idle loop. The purposes of the idle loop are as follows:

1. To schedule power-down traps
2. To schedule control panel, CPU or IPU communication sequences, such as GPR reads and writes, or program counter and PSD reads and writes
3. To schedule IPL sequences (CPU only)
4. To schedule system control panel run commands
5. To execute system control panel reset sequences
6. In the IPU only, to schedule SIPU traps initiated by the CPU SIPU instruction.

2.11.2 Run Indicator

The true state of the RUN indicator denotes that the CPU/IPU is processing software instructions and scheduling traps and interrupts.

The following events cause the CPU or IPU to enter the run state:

1. A power-up auto restart that is error free
2. A power-up automatic IPL that is error free (CPU only)
3. A control panel IPL command that is error free (CPU only)
4. A control panel run command
5. Temporarily, during a control panel, instruction step sequence
6. In the IPU, by the occurrence of a SIPU trap initiated by the CPU SIPU instruction.

The RUN state can be terminated by the halt state as previously described.

2.11.3 Wait Indicator

The WAIT indicator denotes that the CPU/IPU is in a wait state. The WAIT indicator is only on in conjunction with either the run or halt indicator. The occurrence of the wait and halt combination indicates that the CPU/IPU was halted while in the wait state. The occurrence of the run and wait combination indicates that the CPU/IPU is waiting for an interrupt or console attention trap before processing continues.

Note that when the CPU/IPU is interrupted or trapped out of the wait state, the interrupted (or trapped) program status doubleword stored in the interrupt or trap context block points directly at the wait instruction. If the interrupt or trap handler returns to the point of interrupt or trap, the wait instruction will be reexecuted.

The wait state can only be entered by a software wait instruction. The wait state is terminated by an interrupt or trap.

2.11.4 Interrupt Indicator

The interrupt (INTRPT) active indicator denotes that the CPU/IPU is operating with one interrupt level or more in an active condition. The interrupt active indicator is also used during automatic trap halt sequences when an interrupt may or may not be active.

Excluding automatic trap halt, the interrupt active indicator is turned on by one of the following sequences:

1. An interrupt occurs and the handler operates with interrupts unblocked, which as a result of an acknowledge and activate, SelBUS sequence, causes the interrupt level to go active.
2. A software activate interrupt instruction
3. A software activate channel interrupt instruction.

The indicator can be turned off by either a deactivate instruction or a deactivate channel interrupt instruction. Either instruction deactivates the last active level in the system.

Note that when the interrupt active indicator is on, some level of CPU and system interruptability is inhibited.

2.11.5 Parity Error Indicator

The parity error (PE) indicator on the IOP console denotes that the CPU has encountered a memory parity error during instruction execution. The PE indicator turns on when the CPU detects the parity error during an instruction memory read. Parity errors cannot be detected by memory write functions. The PE indicator remains on for the duration of the software parity error trap handler, until the handler executes a load program status doubleword, or a load program status doubleword and change map instruction, which turns off the PE indicator. If a PE trap causes an automatic trap halt, the PE indicator will remain on until a system control panel reset function is executed.

By definition, a parity error is a parity error in a core memory system or an uncorrectable data error in a MOS memory system.

It is normal for the PE indicator to turn on when memory is being initialized during system initialization and software bootstrap sequences.

2.12 Operator Commands

Table 2-12 lists the system console commands (matching the command syntax and definition). Each command syntax and carriage return (CR) combination input to the console causes a decoding of the input characters and execution of the command. Table 2-13 lists the console displays that result from these commands.

The following descriptions include some of the operator commands and their effects on the CPU/IPU operating states and addressing environments. A more extensive description of system operator console functions is available in the CONCEPT 32/67 System Manual, Publication Order Number 312-000410-00.

2.12.1 Halt Command

The halt command causes the processor to exit the run state and enter the halt state. In some cases, the transition from run to halt may not be instantaneous, and both run and halt indicators may turn on for a period not to exceed two seconds.

When the processor halts, a halt display is provided on the panel. The halt display consists of the PSD word 1 and the memory location (instruction) addressed by PSD1. When in the RUN mode, that the address or program counter value in PSD1 is a logical address and must be converted to a physical address through the processor map before the correct memory location can be displayed. The logical to physical memory address conversion is performed automatically by the halt display or any panel read of the processor PSD1.

2.12.2 Run Command

The run command causes the processor to exit the halt state and enter the run state. A run LED is provided on the turnkey panel. A run command may be executed while the CPU is in a run state. The run command causes the IOP to exit panel mode and enter the console mode.

2.12.3 Reset Command

The reset command may only be executed while the processor is in the halt state. This command causes a reset signal to be sent to all devices connected to the SelBUS. The processor then executes a firmware reset.

Note

> If the CPU is in RUN and the IPU is halted, the IOP panel associated with the IPU may execute a reset command which will reset both CPU and IPU.

2.12.4 IPL Command

The IPL command may only be executed while the CPU is in a halt state. The IPL command causes a limited firmware reset and then initiates an I/O initial program load sequence with the IPL device. An earlier IPL description provides the rules for using a default IPL device, or using the IPL device addressed in the panel IPL command. If the IPL command is successful, the CPU will enter the run state, executing the software bootstrap program. However, if IPL errors are encountered, the CPU will execute an automatic trap halt.

Table 2-12
Alphabetical List of System Control Panel Commands

NOTE: [CR] denotes carriage return following the command.
*** denotes a continuation of the current command.

@@A	Attention (when in console mode)
@@C	Enter console mode
@@P	Enter panel mode (when in console mode)
AS[CR]	Clear address stop
AS=XXXXXXXX[CR]	Set address stop
BACKSPACE	Deletes last character input
BAS[CR]	Read base registers
BASA=XXXXXXXX[CR]	Write to base register A
CLE[CR]	Clear memory
[CNTRL][H]	Deletes last character input
CS[CR]	Read control switches
CS=XXXXXXXX[CR]	Set control switches
EA[CR]	Read effective address
GPR[CR]	Read general purpose registers
GPRA=XXXXXXXX[CR]	Write general purpose register A
HALT[CR]	Halt
IPL[CR]	IPL from default address (CPU only)
IPL=XXXX[CR]	IPL from XXXX (CPU only)
IS[CR]	Clear instruction stop
IS=XXXXXXXX[CR]	Set instruction stop
LF	Repeat command (except RST)
MA=XXXXXX[CR]	Read physical memory address location
[CR]	*** increment and read memory address
MAV=XXXXXX[CR]	Read virtual memory address location
[CR]	*** increment and read memory address
MD=XXXXXXXX[CR]	Write memory data
=XXXXXXXX[CR]	*** increment and write memory data
[CR]	*** increment and write previous data
MSGE[CR]	Message
OVR[CR]	Toggle clock override (CPU only)
PC=XXXXXXXX[CR]	Load program counter
PRIP[CR]	Set primary panel (master terminal only)
PSD[CR]	Read program status doubleword (PSD1 and PSD2)
PSD=XXXXXXXX[CR]	Write program status word (PSD2)
PSW[CR]	Read program status word (PSD1)
PSW=XXXXXXXX[CR]	Write program status word (PSD1)
RS[CR]	Clear read operand stop
RS=XXXXXXXX[CR]	Set read operand stop
RST[CR]	Reset
RUN[CR]	Run
SECP[CR]	Set secondary panel (master and slave terminals)
SHIFT DEL	Deletes entire command line (hazeltine terminal only)
STEP[CR]	Instruction step
[CR]	*** instruction step
WS[CR]	Clear write operand stop
WS=XXXXXXXX[CR]	Set write operand stop

Table 2-13
State Indicator Pairs and Listing

State indicators are paired with other state indicators as follows:

Memory address (MA) - Memory data (MD)
Memory address (MA) - Control switches (CS)
Program status word (PSW) - Instruction (INST)
Program status doubleword (PSD) - (Blank)

AS	Address stop set (call three stops set)
CS	Control switch settings is adjacent display (NNNNNNNN)
EA	Effective address in adjacent display (NNNNNNNN)
HALT	CPU or IPU in halt mode
INST	Instruction in adjacent display (NNNNNNNN)
INT	Interrupt active
IS	Instruction stop set
MA	Memory address in adjacent display (NNNNNNNN)
MD	Memory data in adjacent display (NNNNNNNN)
OVR	Clock override on
PE	Parity error or uncorrectable data error from memory
PSD	Program status doubleword in adjacent two displays (NNNNNNNN NNNNNNNN)
PSW	Program status word 1 in adjacent display (NNNNNNNN)
RS	Operand read stop set
RUN	CPU in run mode
ST	Stop. One of the address stop conditions has been set and caused the halt condition
WAIT	Instruction execution not in progress
WS	operand write stop set

Normally, the IPL command should be preceded by a reset command. After the reset, the general-purpose registers may be loaded with the data that is passed through the IPL sequence to the software bootstrap program.

IPL cannot be initiated from the IPU console.

2.12.5 Attention Command

The attention command causes a processor console attention trap signal. If interrupts are unblocked, and if no previous console attention trap is being processed, the trap will fire and the processor will execute a console attention trap. When the console attention trap fires, the processor disables the trap until trap processing (software handler) is complete. The software then executes a LPSD or a LPSDCM instruction which reenables the console attention trap.

If an attention command is executed while software traps are disabled, an automatic trap halt will occur. If an attention command is executed while interrupts are blocked, or a previous console attention trap is being processed, the attention command will remain pending until interrupts are unblocked, or a LPSD/LPSDCM instruction is executed. The attention command may be used while the processor is in either a run state or halt state; however, the console attention trap is only scheduled by the processor when in the run state.

2.12.6 Write PSW Command

The write PSW command is specified by the 'PSW=XXXXXXXX' syntax, where XXXXXXXX is a 32-bit PSD word 1. This command is capable of altering the following:

1. Privileged/unprivileged state (bit 0)
2. Condition codes (bits 1 through 4)
3. Extended/non extended address (bit 5)
4. Base/non base mode (bit 6)
5. Enable/disable arithmetic exception (bit 7)
6. The 19-bit CPU program counter (bits 13 through 31), including the right halfword flag (bit 30).
7. Bit 31 is not used and has no effect.

The write PSW command may only be executed while the processor is halted.

2.12.7 Write PC Command

The write PC command is specified by the 'PC=XXXXX' syntax, where XXXXX is a 24-bit right justified program counter value to be written into the program counter field (bits 13 through 31) of PSD word 1.

The write PC command is used to alter the program counter without altering the state control bits (bits 0 through 7) of PSD word 1. The write PC command may only be used while the processor is halted.

2.12.8 Write PSD2 Command

The write PSD2 command is specified by the 'PSD=XXXXXXXX' syntax, where XXXXXXXX is a 32-bit PSD word 2. This command may be used to alter the following:

1. Set mapped or unmapped environment (bit 0)
2. If mapped (bit 00 = 1), and if bit 15 = 1; retain the map contents and the current process index (CPIX) value.
3. If mapped (bit 00 = 1), and if bit 15 = 0; load the new map, using the CPIX provided by bits 18 through 31.
4. Retain current interrupt blocking state (bit 16 = 1).

5. Block interrupts (bit 17 = 1).

6. Unblock interrupts (bits 16 and 17 = 00).

The current PSD word 2 is monitored by the PSD display. The write PSD2 command may only be used while the processor is halted. In addition, all map pointers required to describe the map contents must be preloaded into memory as described in the memory management discussion of the processor reference manual.

2.12.9 Address Stop Commands

The IOP system control panel/operator console implements four address stop commands that monitor processor-to-memory communications. When the IOP address stop logic detects an address compare of a processor-initiated memory operation (read or write), the IOP sends the processor a SelBUS signal, the system control panel attention (SCPATTN or IPPATTN). The SCPATTN or IPPATTN causes the CPU to exit the run state and enter the halt state.

The specific commands and their syntax used to arm the address stop logic are as follows:

1. 'IS=XXXXXX' which arms the instruction read stop logic
2. 'RS=XXXXXX' which arms the operand read stop logic
3. 'WS=XXXXXX' which arms the operand write stop logic
4. 'AS=XXXXXX' which arms instruction read, operand read, and operand write stop logic.

In each of the above, XXXXXX is a 24-bit word-aligned, real address that is used for the address stop comparison.

The address stop commands may be entered singularly or in combination to achieve a multiple mode address stop; however, the IOP provides only one address holding register and one address comparator. Therefore, only the most recently entered stop address will be retained for the address stop comparison.

The read address stop commands (IS, RS, AS) turn off the CPU cache and IPU cache. The WS command by itself does not turn off the processor cache.

The IOP address stop logic may be cleared singularly by using a 'IS', 'RS', or 'WS' command to clear the corresponding stop function without affecting any other address stop functions that are set. The 'AS' command may be used to globally clear all of the address stop functions.

The stop address that is entered with the address stop command is interpreted as a 24-bit real (physical) address that is word oriented. If the entered address contains halfword or doubleword flags (C-bits) in bits 23 and 24 of the address, these bits will be masked out when the address is loaded into the address stop register. Similarly, the C-bits of a processor memory access are masked out prior to the address stop compare so that any processor access (byte, halfword, word, or doubleword) of the specified word address will cause a processor address stop.

When the address stop commands are used with a 32/67 processor the stop address may specify any address the processor will access, including the dedicated memory addresses used with I/O, trap, interrupt, or map load functions, and the processor will stop. This is a functional difference between the CONCEPT 32/67 and other Gould S.E.L. processors, namely the 32/50-series and the 32/70-series.

Address stop may be set while the processor is in either the halt or run state. The state of the address stop logic (stops that are armed) and the occurrence of address stops are displayed on the IOP system control panel/operator console.

2.12.10 Address Stop Characteristics

The following items describe the operational characteristics of the address stop implemented with the CONCEPT 32/67 processor:

1. Instruction read stops cause the processor to halt prior to the execution of the specified instruction.
2. Operand read or operand write address stops cause the processor to halt after the instruction that caused the specified memory access is completed.
3. Operand read or write stops that specify one word of a doubleword, or one word of a file (8 words), causes the processor to halt after the entire file or the doubleword access is completed. WS is imprecise.
4. If an instruction read stop, and an error trap relating to the fetch that caused the instruction read stop, occur at the same time, the processor will stop, but the PSD will point to the software error trap handler.
5. If an operand read or write stop, and an interrupt or error trap, occur relating to the memory access that caused the operand read or write stop, the processor will stop, but the PSD will point to the software interrupt or error trap handler.
6. The operand read stop may be used to detect the processor access of an indirect word.
7. The operand read stop may be used to detect the processor access of a new PSD, or a map image descriptor, during context switching (trap, interrupt, LPSD, or LPSDCM).
8. The operand write stop may be used to detect the processor store of the old PSD, or the trap status word, during interrupt or trap context switching.
9. The instruction read stop may be used to detect the target of a branch or a branch indirect instruction.
10. The instruction read stop may be used to detect the target of an execute memory or an execute memory indirect instruction.
11. If an instruction read stop is set to a pair of halfword instructions, the processor will stop prior to the execution of both halfword instructions.
12. If an instruction read stop occurs, the run or step commands will cause the processor to execute the specified instruction without stopping a second time. If the specified instruction is accessed a third time, the processor will stop again.
13. If an instruction read stop occurs, and an interrupt becomes pending while the processor is halted; then when the run command is executed, the processor will context switch to the software interrupt handler, and reexecute the instruction stop when the software returns control to the point of interrupt.
14. The address stop logic may not be used to detect a memory access of an I/O channel or SelBUS device other than the processor.

2.12.11 Step Command

The step command, also referred to as an instruction step command, causes the processor to execute one software instruction. The step command, may only be used while the processor is in a halt state.

The following items describe the operational characteristics of the step command:

1. The step command causes the execution of one halfword or one fullword instruction. The post-execution PSD and the next instruction to be executed are displayed on the system control panel/operator console.

2. If the instruction to be stepped is an execute memory or an execute register instruction, both the execute and its target will be stepped as one instruction.

3. If the step command causes an instruction fetch error, the processor will step into the software error trap handler or perform an auto trap halt.

4. If an interrupt is pending when the step command is used, the processor will step into the software interrupt handler.

5. If the step command causes an operand read or write error, the processor will step into the software error trap handler or perform an automatic trap halt.

6. If the step command causes an arithmetic exception error, and the trap is enabled, the processor will halt normally with the arithmetic exception pending. The next step or run command causes the processor to advance into the software arithmetic exception trap handler.

7. If the step command causes an instruction address stop that was not the cause of the current processor halt state, the address stop will be executed and the target instruction will not be executed.

8. If the step command causes an instruction address stop that was the cause of the current processor halt state, the address stop is cleared and ignored and the target instruction will be executed.

9. During a step command, operand read or write address stops are ignored.

10. A step command of a start I/O data transfer instruction addressed to the IOP console should not be performed. If it is performed, the start I/O will become queued up in the IOP channel interface, and the data transfer cannot be executed or terminated until the processor is commanded to the run state, and the IOP system control panel enters the console mode.

2.12.12 Memory Address Virtual Command

The memory address virtual (MAV) command is specified by the syntax 'MAV=XXXXXX', where the XXXXXX is a 24-bit logical address. The IOP sends the logical address to the processor for conversion from logical to physical through the processor's map and memory management hardware. The processor then sends the physical address back to the IOP which displays the physical address and the contents of the specified memory location.

It must be noted that the processor does not check the validity of the logical address during the conversion process and erroneous results can occur if the logical address is greater than the current mapping environment and the map contents allow. The MAV command may only be used while the processor is halted.

2.12.13 Read Effective Address Command

The read effective address (EA) command is specified by the syntax 'EA' and may only be executed while the processor is halted. The EA command causes the processor to fetch the instruction addressed by the PSD and compute the effective logical address of the instruction, after resolving any indexing and indirection specified by the instruction. The effective logical address is then sent to the IOP for display purposes.

For this command to function correctly, the instruction addressed by the PSD must be a memory reference instruction and if it specifies indirect addressing, the indirect chain must be free of memory error.

The effective logical address that is computed will contain both F-bit (bit 12) and C-bits (bits 30 and 31) if the processor is in a nonextended addressing mode. The logical address in this mode is 20 bits. If the processor is in the extended address mode, the F-bit will be omitted, and the logical address is 24 bits in length.

2.13 Instruction Attributes

The instruction attributes together with the opcodes, mnemonics, and firmware vector locations, are listed in table 2-14. Where applicable, the address specification column contains a reference to a rule number associated with the instruction attribute. These rules are listed in table 2-15.

Table 2-14
Instruction Attributes (Sheet 1 of 15)

Opcode	Mnemonic	Firmware Vector	Instruction Attribute	Address Specification
0000	HALT	@820	HALF	N/A
0001	WAIT	@822	HALF	N/A
0002	NOP	@824	HALF	N/A
0003	LCS	@826	HALF, MODREG, LOCKLMAR	N/A
0004	ES	@828	HALF, MODREG	N/A
0005	RND	@82A	HALF, MODREG	N/A
0006	BEI	@82C	HALF	N/A
0007	UEI	@82E	HALF	N/A
0008	EAE	@830	HALF	N/A
0009	RDSTS	@832	HALF, MODREG	N/A
000A	SIPU	@834	HALF	N/A
000B	RWCS	@836	HALF, LOCKLMAR	N/A
000C	WWCS	@838	HALF, LOCKLMAR	N/A
000D	SEA	@83A	HALF, LOCKLMAR	N/A
000E	DAE	@83C	HALF	N/A
000F	CEA	@83E	HALF, LOCKLMAR	N/A
0400	ANR	@840	HALF, MODREG	N/A
0407	SMC	@846	HALF, LOCKLMAR	N/A
040A	CMC	@844	HALF, LOCKLMAR	N/A
040B	RPSWT	@848	HALF, MODREG	N/A
0800	ORR	@850	HALF, MODREG	N/A
0808	ORRM	@858	HALF, MODREG	N/A
0C00	EOR	@860	HALF, MODREG	N/A
0C08	EORM	@868	HALF, MODREG	N/A
1000	CAR	@870	HALF	N/A
1400	CMR	@880	HALF	N/A
1800	SBR	@884	HALF, LOCKLMAR, MODREG	N/A
1C00	ZBR	@888	HALF, LOCKLMAR, MODREG	N/A
2000	ABR	@88C	HALF, LOCKLMAR, MODREG	N/A
2400	TBR	@890	HALF	N/A
2800	TRSW	@8E8	HALF, LOCKLMAR	N/A
2C00	TRR	@900	HALF, MODREG	N/A
2C03	TRC	@906	HALF, MODREG	N/A
2C04	TRN	@908	HALF, MODREG	N/A
2C05	XCR	@90A	HALF, LOCKLMAR, MODREG	N/A
2C07	LMAP	@90C	HALF	N/A
2C08	TRRM	@90E	HALF, MODREG	N/A
2C09	SETCPU	@910	HALF	N/A
2C0A	TMAPR	@912	HALF, MODREG	N/A
2C0B	TRCM	@914	HALF, MODREG	N/A
2C0C	TRNM	@916	HALF, MODREG	N/A
2C0D	XCRM	@918	HALF, LOCKLMAR, MODREG	N/A

Table 2-14
Instruction Attributes (Sheet 2 of 15)

Opcode	Mnemonic	Firmware Vector	Instruction Attribute	Address Specification
2C0E	TRSC	@91A	HALF	N/A
2C0F	TSCR	@91C	HALF, MODREG	N/A
340X	LA	@B3A	INVALID RWH, MODREG, LAORLEA, MEMREF, FULLWORD	N/A
3800	ADR	@920 *@940	HALF, MODREG	N/A
3801	ADRFW	@922 *@942	HALF, MODREG	N/A
3803	SURFW	@926 *@946	HALF, MODREG	N/A
3804	DVRFW	@928 *@948	HALF, MODREG	N/A
3805	FIXW	@92A *@94A	HALF, MODREG	N/A
3806	MPRFW	@92C *@94C	HALF, MODREG	N/A
3807	FLTW	@92E *@94E	HALF, MODREG	N/A
3808	ADRM	@930 *@950	HALF, MODREG	N/A
3809	ADRFD	@932 *@952	HALF, DMODREG	Rule 4
380B	SURFD	@936 *@956	HALF, DMODREG	Rule 4
380C	DVRFD	@938 *@958	HALF, DMODREG	Rule 4
380D	FIXD	@93A *@95A	HALF, DMODREG	Rule 4
380E	MPRFD	@93C *@95C	HALF, DMODREG	Rule 4
380F	FLTD	@93E *@95E	HALF, DMODREG	Rule 4
3C00	SUR	@969	HALF, MODREG	N/A
3C08	SURM	@964	HALF, MODREG	N/A
4000	MPR	@924 *@944	HALF, DMODREG	Rule 5
4400	DVR	@934 *@954	HALF, DMODREG	Rule 5
6000	NOR	@9A0	HALF, LOCKLMAR, MODREG	N/A
6400	NORD	@9B0	HALF, LOCKLMAR, MODREG	N/A
6800	SCZ	@874	HALF, LOCKLMAR, MODREG	N/A
6C00	SRA	@8A0	HALF, MODREG	N/A
6C40	SLA	@8A8	HALF, MODREG	N/A

* HARDWARE FLOATING-POINT VECTOR

Table 2-14
Instruction Attributes (Sheet 3 of 15)

Opcode	Mnemonic	Firmware Vector	Instruction Attribute	Address Specification
7000	SRL	@8A4	HALF, MODREG	N/A
7060	SLL	@8B4	HALF, MODREG	N/A
7400	SRC	@8E0	HALF, MODREG	N/A
7440	SLC	@8E4	HALF, MODREG	N/A
7800	SRAD	@8C0	HALF, DMODREG	Rule 5
7840	SLAD	@8C8	HALF, DMODREG	Rule 5
7C00	SRLD	@8C4	HALF, DMODREG	Rule 5
7C40	SLLD	@8D2	HALF, DMODREG	Rule 5
800X	LEAR	@B3C	FULLWORD, INVALID RHW, LOCKLMAR, MODREG, LEAR, MEMREF	N/A
8400	ANMW	@9C0	FULLWORD, INVALID RHW, MODREG, MEMREAD,MEMREF	Rule 1
8400	ANMH	@9C1	FULLWORD, INVALID RHW, MODREG, MEMREAD, MEMREF	Rule 1
8400	ANMD	@9C2	FULLWORD, INVALID RHW, MODREG, MEMREAD, MEMREF	Rule 1
8408	ANMB	@9C4	FULLWORD, INVALID RHW, MODREG, MEMREAD, MEMREF	Rule 1
8800	ORMW	@9D0	FULLWORD, INVALID RHW, MODREG, MEMREAD, MEMREF	Rule 1
8800	ORMH	@9D1	FULLWORD, INVALID RHW, MODREG, MEMREAD, MEMREF	Rule 1
8800	ORMD	@9D2	FULLWORD, INVALID RHW, MODREG, MEMREAD, MEMREF	Rule 1
8808	ORMB	@9D4	FULLWORD, INVALID, RHW, MODREG, MEMREAD, MEMREF	Rule 1
8C00	EOMW	@9E0	FULLWORD, INVALID RHW, MODREG, MEMREAD, MEMREF	Rule 1
8C00	EOMH	@9E1	FULLWORD, INVALID RHW, MODREG, MEMREAD, MEMREF	Rule 1
8C00	EOMD	@9E2	FULLWORD, INVALID RHW, MODREG, MEMREAD, MEMREF	Rule 1

Table 2-14
Instruction Attributes (Sheet 4 of 15)

Opcode	Mnemonic	Firmware Vector	Instruction Attribute	Address Specification
8C08	EDMB	@9E4	FULLWORD, INVALID RHW, MODREG, MEMREAD, MEMREF	Rule 1
9000	CAMW	@9F0	FULLWORD INVALID RHW, MEMREAD, MEMREF	Rule 1
9000	CAMH	@9F1	FULLWORD INVALID RHW, MEMREAD, MEMREF	Rule 1
9000	CAMD	@9F2	FULLWORD INVALID RHW, MEMREAD, MEMREF	Rule 1
9008	CAMB	@9F4	FULLWORD, INVALID RHW, MEMREAD, MEMREF	Rule 1
9400	CMMW	@A00	FULLWORD, INVALID RHW, MEMREAD, MEMREF	Rule 1
9400	CMMH	@A01	FULLWORD, INVALID RHW, MEMREAD, MEMREF	Rule 1
9400	CMMD	@A02	FULLWORD, INVALID RHW, MEMREAD, MEMREF	Rule 1
9408	CMMB	@A04	FULLWORD, INVALID RHW, MEMREAD, MEMREF	Rule 1
980X	SBM	@A10	MEMWRT, READCHECK-STORED LOCKLMAR, MEMREAD	Rule 6
9C0F	ZBM	@A18	FULLWORD, INVALID RHW, FRCWORD, MEMREF	Rule 6
A00X	ABM	@A20	FULLWORD, INVALIDRHW, MEMREF, FRCWORD, MEMWRT, MEMREAD, READCHECKSTORE, LOCKLMAR	Rule 6
A40X	TBM	@A28	FULLWORD, INVALIDRHW, FRCWORD, MEMREF, MEMREAD	Rule 6

Table 2-14
Instruction Attributes (Sheet 5 of 15)

Opcode	Mnemonic	Firmware Vector	Instruction Attribute	Address Specification
A80X	EXM	@A30	FULLWORD, INVALIDRHW, FRCWORD, MEMREAD, MEMREF, LOCKLMAR, DMODREG, MODREG	Rule 6
AC0X	LW	@A50	FULLWORD, INVALIDRHW, MODREG, MEMREAD, MEMREF	Rule 1
AC0X	LH	@A51	FULLWORD, INVALIDRHW, MODREG, MEMREAD, MEMREF	Rule 1
AC0X	LD	@A52	FULLWORD, INVALIDRHW, MODREG, MEMREAD, MEMREF	Rule 1
AC08	LB	@A54	FULLWORD, INVALIDRHW, MODREG, MEMREAD, MEMREF	Rule 1
B00X	LMW	@A70	FULLWORD, INVALIDRHW, MODREG, MEMREAD, MEMREF	Rule 1
B00X	LMH	@A71	FULLWORD, INVALIDRHW, MODREG, MEMREAD, MEMREF	Rule 1
B00X	LMD	@A72	FULLWORD, INVALIDRHW, MODREG, MEMREAD, MEMREF	Rule 1
B008	LMB	@A74	FULLWORD, INVALIDRHW, MODREG, MEMREAD, MEMREF	Rule 1
B40X	LNW	@A90	FULLWORD, INVALIDRHW, MODREG, MEMREAD, MEMREF	Rule 1
B40X	LNH	@A91	FULLWORD, INVALIDRHW, MODREG, MEMREAD, MEMREF	Rule 1

Table 2-14
Instruction Attributes (Sheet 6 of 15)

Opcode	Mnemonic	Firmware Vector	Instruction Attribute	Address Specification
B40X	LND	@A92	FULLWORD, INVALIDRHW, MODREG, MEMREAD, MEMREF	Rule 1
B408	LNB	@A94	FULLWORD, INVALIDRHW, MODREG, MEMREAD, MEMREF	Rule 1
B80X	ADNW	@AB0	FULLWORD, INVALIDRHW, MODREG, MEMREAD, MEMREF	Rule 1
B80X	ADMH	@AB1	FULLWORD, INVALIDRHW, MODREG, MEMREAD, MEMREF	Rule 1
B80X	ADMD	@AB2	FULLWORD, INVALIDRHW, MODREG, MEMREAD, MEMREF	Rule 1
B808	ADMB	@AB4	FULLWORD, INVALIDRHW, MODREG, MEMREAD, MEMREF	Rule 1
BC0X	SUMW	@AD0	FULLWORD, INVALIDRHW, MODREG, MEMREAD, MEMREF	Rule 1
BC0X	SUMH	@AD1	FULLWORD, INVALIDRHW, MODREG, MEMREAD, MEMREF	Rule 1
BC0X	SUMD	@AD2	FULLWORD, INVALIDRHW, MODREG, MEMREAD, MEMREF	Rule 1
BC08	SUMB	@AD4	FULLWORD, INVALIDRHW, MODREG, MEMREAD, MEMREF	Rule 1
C00X	MPMW	@A40 +@A60	FULLWORD, INVALIDRWH, DMODREG, MEMREAD, MEMREF, LOCKLMAR	Rule 1

+ HARDWARE FIXED-POINT VECTOR

Table 2-14
Instruction Attributes (Sheet 7 of 15)

Opcode	Mnemonic	Firmware Vector	Instruction Attribute	Address Specification
C00X	MPMH	@A41 +@A61	FULLWORD, INVALIDRHW, DMODREG, MEMREAD, MEMREF, LOCKLMAR	Rule 1
C008	MPMB	@A44 +@A64	FULLWORD, INVALIDRHW, DMODREG, MEMREAD, MEMREF, LOCKLMAR	Rule 1
C40X	DVMW	@AF0	FULLWORD, INVALIDRHW, DMODREG, MEMREAD, MEMREF	Rule 1
C40X	DVMH	@AF1	FULLWORD, INVALIDRHW, DMODREG, MEMREAD, MEMREF	Rule 1
C408	DVMB	@AF4	FULLWORD, INVALIDRHW, DMODREG, MEMREAD, MEMREF	Rule 1
C800	LI	@A80 +@AA0	FULLWORD, INVALIDRHW, MODREG	N/A
C801	ADI	@A82 +AA2	FULLWORD, INVALIDRHW, MODREG	N/A
C802	SUI	@A84 +@AA4	FULLWORD, INVALIDRHW, MODREG	N/A
C803	MPI	@A86 +@AA6	FULLWORD, INVALIDRHW, DMODREG	Rule 5
C804	DVI	@A88 +@AA8	FULLWORD, INVALIDRHW, DMODREG	Rule 5
C805	CI	@A8A +@AAA	FULLWORD, INVALIDRHW	N/A
C806	SVC	@A8C +@AAC	FULLWORD, INVALIDRHW, LOCKLMAR	N/A

+ HARD FIXED-POINT VECTOR

Table 2-14
Instruction Attributes (Sheet 8 of 15)

Opcode	Mnemonic	Firmware Vector	Instruction Attribute	Address Specification
C807	EXR	@A8E +@AAE	FULLWORD, INVALIDRHW	N/A
CC00 - CC07	LF	@B10-B17	MEMREF, MEMREAD, ZEROFB	Rule 3
CC08 - CC0F	LFBR	@B18-B1F	FULLWORD, INVALIDRHW, LOCKLMAR, MODREG	Rule 3
D00X	LEA	@B3E	FULLWORD, INVALIDRHW, MODREG, MEMREF, LAORLEA	N/A
D40X	STW	@AC0	FULLWORD, INVALIDRHW, LOCKLMAR MODREG, MEMWRT STORECKMAP, MEMREF	Rule 1
D40X	STH	@AC1	FULLWORD, INVALIDRHW, LOCKLMAR MODREG, MEMWRT STORECKMAP, MEMREF	Rule 1
D40X	STD	@AC2	FULLWORD, INVALIDRHW, LOCKLMAR MODREG, MEMWRT STORECKMAP, MEMREF	Rule 1
DX408	STB	@AC4	FULLWORD, INVALIDRHW, MODREG, MEMWRT STORECKMAP, MEMREF	Rule 1
D80X	STMW	@B00	FULLWORD, INVALIDRHW, LOCKLMAR, MEMWRT, STORECKMAP, MEMREF	Rule 1

Table 2-14
Instruction Attributes (Sheet 9 of 15)

Opcode	Mnemonic	Firmware Vector	Instruction Attribute	Address Specification
D80X	STMH	@B01	FULLWORD, INVALIDRHW, LOCKLMAR, MEMWRT, STORECKMAP, MEMREF	Rule 1
D80X	STDM	@BO2	FULLWORD, INVALIDRHW, LOCKLMAR, MEMWRT, STORECKMAP, MEMREF	Rule 1
D808	STMB	@BO4	FULLWORD INVALIDRHW, LOCKLMAR, MEMWRT, STORECKMAP, MEMREF	Rule 1
DC00 - DC07	STF	@B40	FULLWORD, INVALIDRHW, LOCKLMAR, MEMREF,MEMREF, STORECKMAP, MEMWRT, ZEROFB	Rule 3
DC08 - DC0F	STFBF	@B50	FULLWORD, INVALIDRHW, LOCKLMAR, MEMREF,MEMREF, SOTRECKMAP, MEMWRT, ZEROFB	Rule 3
E000	SUFW	@B80 *@BA0	FULLWORD, INVALIDRHW, MODREG, MEMREF, LOCKLMAR, MEMREAD, ZEROFB	Rule 6
E000	SUFD	@B82 *@BA2	FULLWORD, INVALIDRHW, MODREG, MEMREF, LOCKLMAR, MEMREAD, ZEROFB	Rule 6

* HARDWARE FLOATING-POINT VECTOR

Table 2-14
Instruction Attributes (Sheet 10 of 15)

Opcode	Mnemonic	Firmware Vector	Instruction Attribute	Address Specification
E008	ADFW	@B90 *@BB0	FULLWORD, INVALIDRHW, MODREG, MEMREF, LOCKLMAR, MEMREAD, ZEROFB	Rule 6
E008	ADFD	@B92 *@BB2	FULLWORD, INVALIDRHW, MODREG, MEMREF, LOCKLMAR, MEMREAD, ZEROFB	Rule 6
E400 - E407	DVFW	@BC0 *@BE0	FULLWORD, INVALIDRHW, MODREG, MEMREF, LOCKLMAR, MEMREAD, ZEROFB	Rule 6
E400 - E407	DVFD	@BC2 *@BE2	FULLWORD, INVALIDRHW, MODREG, MEMREF, LOCKLMAR, MEMREAD, ZEROFB	Rule 6
E408 - E40F	MPFW	@BD0 *@BF0	FULLWORD, INVALIDRHW MODREG, MEMREF, LOCKLMAR, MEMREAD, ZEROFB	Rule 6
E408 - E40F	MPFD	@BD2 *@BF2	FULLWORD, INVALIDRHW MODREG, MEMREF, LOCKLMAR, MEMREAD, ZEROFB	Rule 6
E80X	AREMW	@B30	FULLWORD, INVALIDRHW, READCHECKSTORE, LOCKLMAR, MEMREAD, MEMWRT, MEMREF	Rule 1
E80X	ARMH	@B31	FULLWORD, INVALIDRHW, READCHECKSTORE, LOCKLMAR, MEMREAD, MEMWRT, MEMREF	Rule 1

* HARDWARE FLOATING-POINT VECTOR

Table 2-14
Instruction Attributes Sheet (11 of 15)

Opcode	Mnemonic	Firmware Vector	Instruction Attribute	Address Specification
E80X	ARMD	@B32	FULLWORD, INVALIDRHW, READCHECKSTORE, LOCKLMAR, MEMREAD, MEMWRT, MEMREF	Rule 1
E808	ARMB	@B34	FULLWORD, INVALIDRHW, READCHECKSTORE, LOCKLMAR, MEMREAD, MEMWRT, MEMREF	Rule 1
EC00	BU	@C00	FULLWORD, INVALIDRHW, FRCWORD, BRANCH, MEMREAD, MEMREF	Rule 7
ECXX	BCT	@C02	FULLWORD INVALIDRHW, FRCWORD, BRANCH, MEMREAD, MEMREF	Rule 7
F000	BFT	@C12	FULLWORD, INVALIDRHW, FRCWORD, BRANCH, MEMREAD, MEMREF	Rule 7
F0XX	BCF	@C10	FULLWORD, INVALIDRHW, FRCWORD, BRANCH, MEMREAD, MEMREF	Rule 7
F400	BIB	@C20	FULLWORD, INVALIDRHW, MODREG, FRCWORD, MEMREF, MEMREAD, I3NOINDEX, BRANCH	Rule 7
F420	BIH	@C24	FULLWORD, INVALIDRHW, MODREG, FRCWORD, MEMREF, MEMREAD, I3NOINDEX, BRANCH	Rule 7
F440	BIWH	@C28	FULLWORD, INVALIDRHW, MODREG, FRCWORD, MEMREF, MEMREAD, I3NOINDEX, BRANCH	Rule 7

Table 2-14
Instruction Attributes (Sheet 12 of 15)

Opcode	Mnemonic	Firmware Vector	Instruction Attribute	Address Specification
F460	BID	@C2A	FULLWORD, INVALIDRHW MODREG, FRCWORD, MEMREF, MEMREAD, I3NOINDEX, BRANCH	Rule 7
F80X	ZMH	@C41	FULLWORD INVALIDRHW, LOCKLMAR, STORECHECKMAP, MEMREF, MEMREAD, INHIBIT DOUBLE MEMWRT	Rule 1
F80X	ZMW	@C40	FULLWORD, INVALIDRHW, LOCKLMAR, STORECHECKMAP, MEMREF, MEMREAD, INHIBIT DOUBLE MEMWRT	Rule 1
F80X	ZMD	@C42	FULLWORD, INVALIDRHW, LOCKLMAR, STORECHECKMAP, MEMREF, MEMREAD, INHIBIT, DOUBLE MEMWRT	Rule 1
F808	ZMB	@C44	FULLWORD, INVALIDRHW, LOCKLMAR, STORECHECKMAP, MEMREF, MEMREAD, INHIBIT DOUBLE MEMWRT	Rule 1
F880	BL	@C46	FULLWORD, INVALIDRHW, INHIBIT, DOUBLE FRCWORD, MEMREF, MEMREAD, BRANCH, MEMWRT	Rule 7

Table 2-14
Instruction Attributes (Sheet 13 of 15)

Opcode	Mnemonic	Firmware Vector	Instruction Attribute	Address Specification
F980	LPSD	@C48	FULLWORD, INVALIDRHW, LOCKLMAR, FRCWORD	Rule 2
FA0X	JWCS	@C4C	FULLWORD, INVALIDRHW, LOCKLMAR, MEMREF	N/A
FA80	LPSDCM	@C4A	FULLWORD, INVALIDRHW, LOCKLMAR, FRCWORD	Rule 2
FC00	EI	@C60	FULLWORD, INVALIDRHW, LOCKLMAR	N/A
FC01	DI	@C62	FULLWORD, INVALIDRHW, LOCKLMAR	N/A
FC02	RI	@C64	FULLWORD, INVALIDRHW, LOCKLMAR	N/A
FC03	AI	@C66	FULLWORD, INVALIDRHW, LOCKLMAR	N/A
FC04	DAI	@C68	FULLWORD, INVALIDRHW, LOCKLMAR	N/A
FC05	TD	@C6A	FULLWORD, INVALIDRHW, LOCKLMAR	N/A
FC06	CD	@C6C	FULLWORD, INVALIDRHW, LOCKLMAR	N/A
FC17	SIO	@C70	FULLWORD, INVALIDRHW, LOCKLMAR	N/A

Table 2-14
Instruction Attributes (Sheet 14 of 15)

Opcode	Mnemonic	Firmware Vector	Instruction Attribute	Address Specification
FC1F	TIO	@C70	FULLWORD, INVALIDRHW LOCKLMAR	N/A
FC37	HIO	@C70	FULLWORD, INVALIDRHW, LOCKLMAR	N/A
Base Mode Only				
1008	SCZ	@874	HALF, LOCKLMAR, MODREG	N/A
1800	SBR	@884	HALF, LOCKLMAR, MODREG	N/A
1804	ZBR	@888	HALF, LOCKLMAR, MODREG	N/A
1808	ABR	@88C	HALF, LOCKLMAR, MODREG	N/A
180C	TBR	@890	HALF	N/A
1C00	SRA	@8A0	HALF, MODREG	N/A
1C20	SRL	@8A4	HALF, MODREG	N/A
1C40	SLA	@8A8	HALF, MODREG	N/A
1C60	SLL	@8B4	HALF, MODREG	N/A
2000	SRAD	@8C0	HALF, DMODREG	Rule 5
2020	SRLD	@8C4	HALF, DMODREG	Rule 5
2040	SLAD	@8C8	HALF, DMODREG	Rule 5
2060	SLLD	@8D2	HALF, DMODREG	Rule 5
2400	SRC	@8E0	HALF, MODREG	N/A
2440	SLC	@8E4	HALF, MODREG	N/A
2802	XCBR	@8EC	HALF, LOCKLMAR, MODREG	N/A
2804	TCG	@8F0	HALF, MODREG	N/A
2805	TGCC	@8F2	HALF	N/A
2808	CALL/ BSUB	@8F4	HALF, LOCKLMAR	N/A
280C	TPCBR	@8F8	HALF, LOCKLMAR	N/A
2803	RETURN	@8FA	HALF, LOCKLMAR	N/A
2C01	TRBR	@902	HALF, LOCKLMAR	N/A
2C02	TBRR	@904	HALF, MODREG	N/A

Table 2-14
Instruction Attributes (Sheet 15 of 15)

Opcode	Mnemonic	Firmware Vector	Instruction Attribute	Address Specification
3802	MPR	@924 *@944	HALF, DMODREG	Rule 5
380A	DVR	@934 *@954	HALF, DMODREG	Rule 5
500X	LA	@B3A	FULLWORD, INVALIDRHW, MODREG, MEMREF, LAORLEA	N/A
540X	STWBR	@970	FULLWORD, INVALIDRHW, LOCKLMAR, ZEROFB, MEMREF, STORECKMAP, MEMWRT	Rule 3
5800 - 5807	SUABR	@980	FULLWORD, INVALIDRHW, LOCKLMAR,	N/A
5808 - 580F	LABR	@988	MODREG, ZEROFB, LAORLEA, MEMREF	N/A
5C00 - 5C07	LWBR	@990	FULLWORD, INVALIDRHW, LOCKLMAR, MODREG, MEMREF, MEMREAD, ZEROFB	Rule 3
5C08 - 5C0F	CALM/ BSUBM	@99C	FULLWORD, INVALIDRHW, MODREG, MEMREF, MEMREAD, ZEROFB, LOCKLMAR	Rule 3

* HARDWARE FLOATING-POINT VECTOR

Table 2-15
Address Specification Rules (Sheet 1 of 2)

Rule Number	Qualifier or Type Instruction	Definitions of Conditions That Would Generate Errors
1	Memory Reference	If a doublword instruction is used and the effective address bit (29) is equal to 1 (base/nonbase)
		If a doubleword and the odd register address bit (8) is equal to 1 (base/nonbase)
		If in base mode and the instruction F and C bits do not equal to the F and C effective address (used when indexing is modifying the opcode)
		If in base mode and a doubleword and the effective address bit (29) is equal to 1
		If in base mode and a doubleword and the odd register address bit (8) is equal to 1
	Multiply or divide a memory word	If a doubleword modify register instruction and the odd register address bit (8) is equal to 1 (base/nonbase)
2	Effective address F and C bits are not equal to zero	If F bit is equal to 1 (base/nonbase)
	LPSD	If F bit is equal to zero and effective address bit (30) is equal to 1 (base/nonbase)
	LPSDCM	If F bit is equal to 0 and the effective address bit (31) is equal to 1 (base/nonbase)
		If instruction F and C bits are not equal to the F and C effective address (base mode)
3	Effective address C bit not equal to zero	If effective address bit (30) is equal to 1 (base/nonbase)
		If instruction F and C bits are not equal to the F and C effective address
4	Register to Register double	If instruction bit (8) is equal to 1
5	Register to register double shift (precision)	If instruction bit (11) is equal to 1

Table 2-15
Address Specification Rules (Sheet 2 of 2)

Rule Number	Qualifier or Type Instruction	Definitions of Conditions That Would Generate Errors
6	Bit in memory	If a doubleword and the effective address bit (29) is equal to 1 and it is not an execute memory instruction (floating point only)
		If a doubleword and the odd register address bit (8) is equal to 1 and it is not an execute memory instruction (floating point only)
	Execute Memory	If instruction F and C bits are not equal to the F and C bit effective address and it is not an execute memory instruction (base mode floating point only)
	Floating Point Memory Reference	If it is a halfword instruction and not an execute memory (floating point only)
		If it is an execute memory, the F bit flag is cleared and the effective F bit is equal to 1 (execute memory only)
		If it is not an execute memory, the F bit flag is not cleared and the effective F bit is equal to 0 (bit in memory only)

CHAPTER 3

THEORY OF OPERATION

3.1 Introduction

This chapter contains the theory of operation for the Gould CONCEPT 32/67 CPU. The introductory portion of this chapter provides a functional overview of the CPU. A comprehensive description for each of the units (MS unit, IE unit, CS unit) follows the overview. The latter portion of the chapter contains detailed descriptions of the instruction pipeline, cache, and memory map. References are made to logic drawing sheet numbers in certain areas of text and diagrams. These references are included as supplemental information for Gould S.E.L. field service personnel.

3.2 Overview

The CPU overview describes the functions, data structure, and data sources for each of the three units comprising the CPU. Figure 3-1 depicts the functional interrelationships of the three units and provides a brief description of the major logic elements contained on each board. Figure 3-2 is an overall block diagram of the CPU.

3.2.1 MS Unit

The MS unit contains the following logic circuitry:

- Micro PC address selection logic
- Cache data out logic
- Scratchpad logic
- Control store logic
- Order structure
- Turnkey panel interface

The text that follows for each portion of the logic should be used in conjunction with figure 3-3, the MS Unit Functional Diagram.

3.2.1.1 Micro PC Address Selection Logic

Functions

1) Decodes macroinstructions into uPC addresses
2) Provides micro interrupt control
3) Allows address selection

3.2.1.2 Cache Data Out Logic

Functions

1) Used to hold and output the cache data after operand reads.

3.2.1.3 Scratchpad Logic

Functions

1) Provides an emulation work area and temporary storage for control panel functions, I/O device entries, and interrupt structure information.

3.2.1.4 Control Store Logic

Functions

1) Contains and outputs microwords
2) Allows for Alterable Control Storage (ACS)
3) Allows for Writable Control Storage (WCS)

3.2.1.5 Order Structure

Functions

1) Decodes microword bits into individual output control signals for both CROM and CREG cycle orders.

3.2.1.6 Turnkey Panel Interface

Functions

1) Decodes turnkey panel PROCESSOR SELECT switch
2) Provides driving, buffering, and latching for inputs and indicators to/from the IOP and turnkey panel.

3.2.2 IE Unit

The IE Unit consists of the following circuitry:

- Instruction pipeline
- Set-up logic
- Execution logic
- Test structure
- Order structure

The text that follows for each portion of the logic should be used in conjunction with figure 3-4, the IE Unit Functional Diagram.

3.2.2.1 Instruction Pipeline

Functions

1) Performs instruction decode
2) Provides halfword istruction detection and metering
3) Provides for indirect sequencing
4) Performs CROM and CREG cycle functions

3.2.2.2 Set-up Logic

Functions

1) Performs effective address calculations
2) Performs operand prefetching
3) Generates Logical Memory Addresses

3.2.2.3 Execution Logic

Functions

1) Executes Macroinstructions
2) Performs Arithmetic/logic operations

3.2.2.4 Test Structure

Functions

1) Monitors external input signals to the CPU
2) Monitors internal conditions of the CPU
3) Performs single or multiple signal tests

3.2.2.5 Order Structure

Functions

1) Initializes or terminates internal control signals
2) Serves as qualifiers for certain events
3) Registers strobe signals

3.2.3 CS Unit

The CS unit contains the following logic circuitry:

- SelBUS interface logic
- Operand effective/instruction logical address logic
- Memory management logic
- Cache memory logic

The text that follows for each portion of the logic should be used in conjunction with figure 3-5, the CS Unit Functional Diagram.

3.2.3.1 SelBUS Interface Logic

Functions

1) Monitors SelBUS for externally initiated memory writes
2) Establishes SelBUS transfer priority
3) Drives memory and I/O transfer to SelBUS
4) Monitors SelBUS for response signals
5) Receives SelBUS DRT transfers
6) Monitors for error response signals

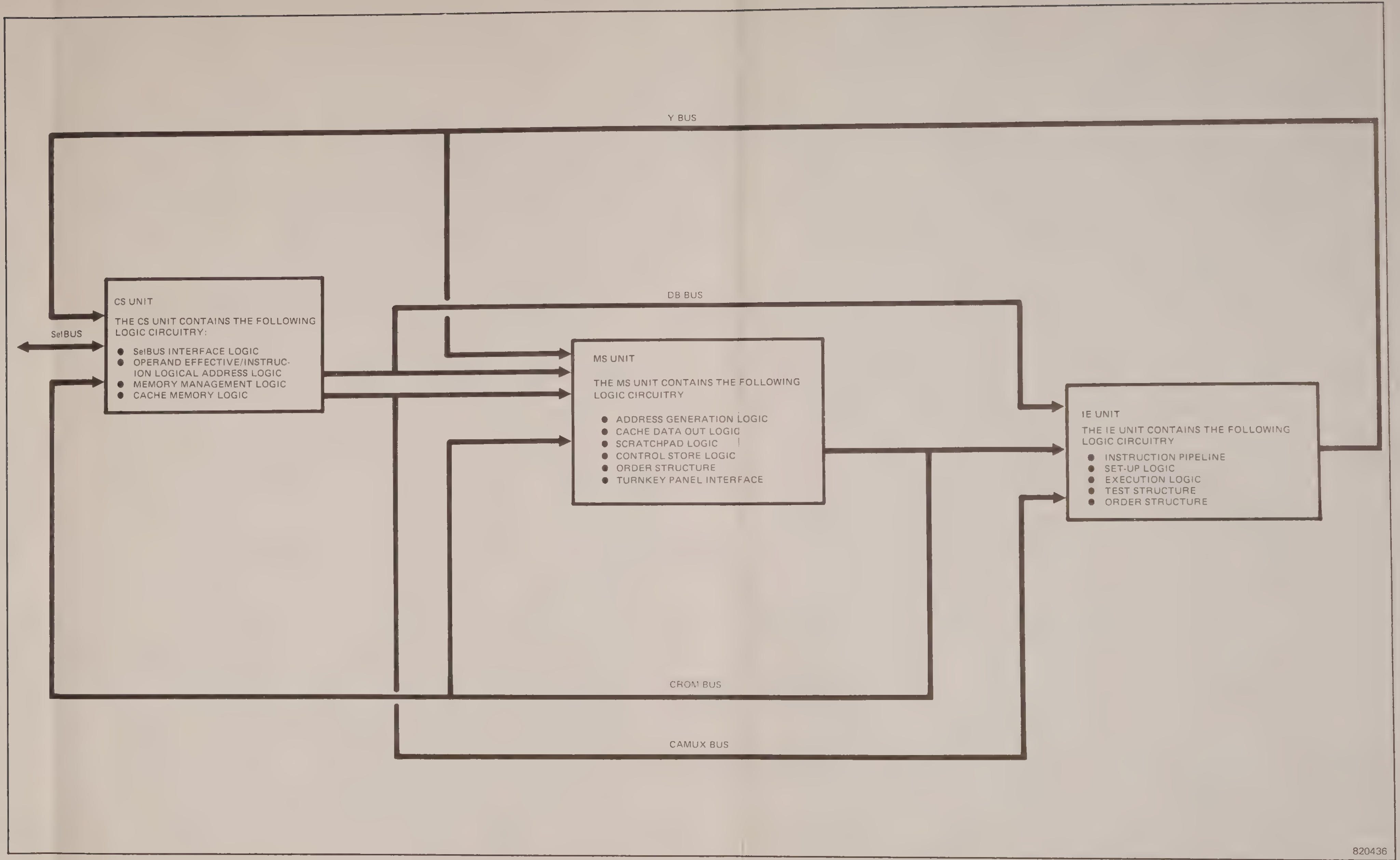

Y BUS
DB BUS
CS UNIT
THE CS UNIT CONTAINS THE FOLLOWING LOGIC CIRCUITRY:
SelBUS
● SelBUS INTERFACE LOGIC
● OPERAND EFFECTIVE/INSTRUCTION LOGICAL ADDRESS LOGIC
● MEMORY MANAGEMENT LOGIC
● CACHE MEMORY LOGIC
MS UNIT
THE MS UNIT CONTAINS THE FOLLOWING LOGIC CIRCUITRY
● ADDRESS GENERATION LOGIC
● CACHE DATA OUT LOGIC
● SCRATCHPAD LOGIC
● CONTROL STORE LOGIC
● ORDER STRUCTURE
● TURNKEY PANEL INTERFACE
IE UNIT
THE IE UNIT CONTAINS THE FOLLOWING LOGIC CIRCUITRY
● INSTRUCTION PIPELINE
● SET-UP LOGIC
● EXECUTION LOGIC
● TEST STRUCTURE
● ORDER STRUCTURE
CROM BUS
CAMUX BUS
820436

Figure 3-1. 32/67 CPU Functional Diagram

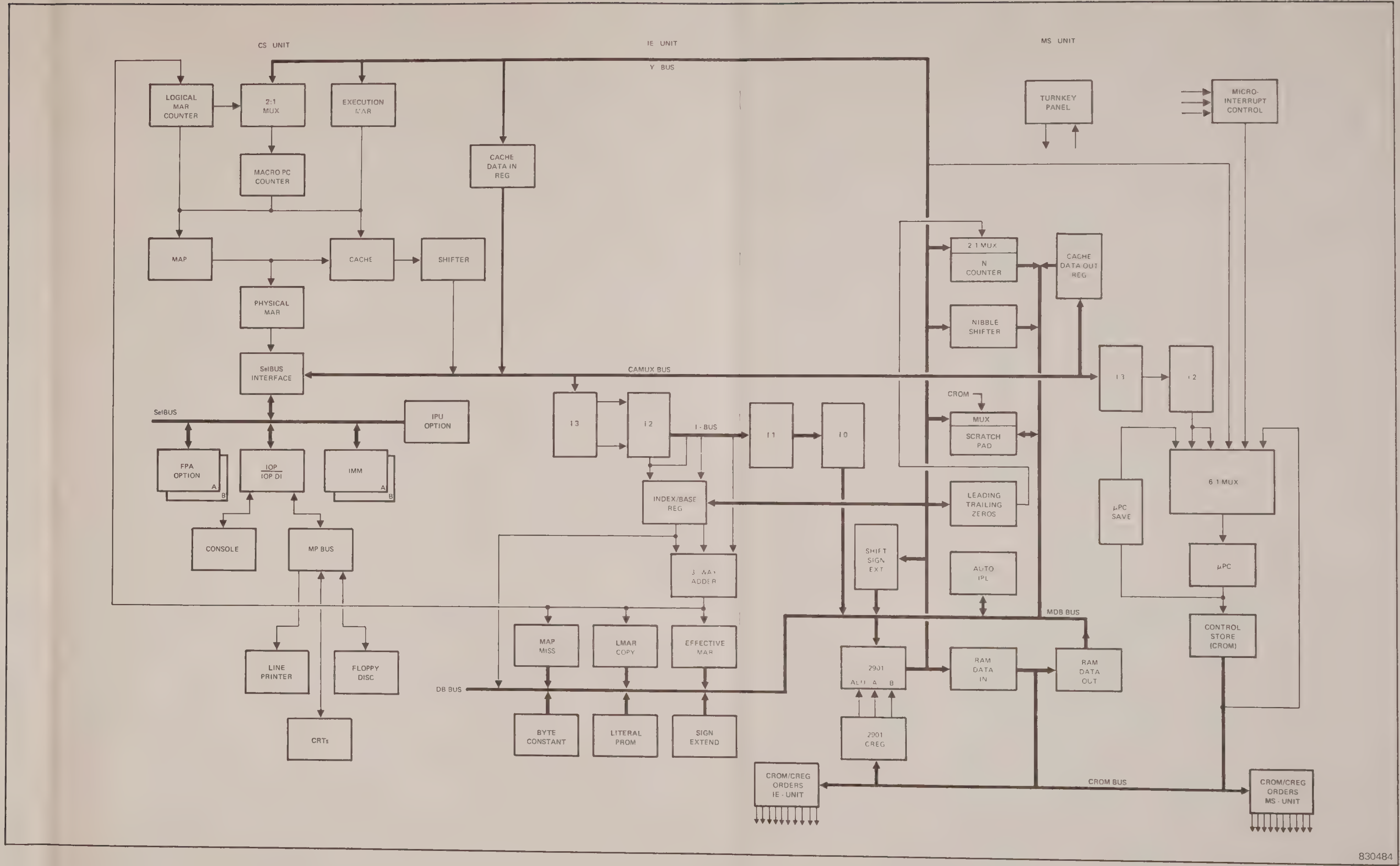

CS UNIT
IE UNIT
MS UNIT
Y BUS
LOGICAL MAR COUNTER
2:1 MUX
EXECUTION MAR
MACRO PC COUNTER
CACHE DATA IN REG
MAP
CACHE
SHIFTER
PHYSICAL MAR
SelBUS INTERFACE
CAMUX BUS
SelBUS
IPU OPTION
FPA OPTION
A
B
IOP
IOP DI
IMM
A
B
CONSOLE
MP BUS
LINE PRINTER
FLOPPY DISC
CRTs
I3
I2
I - BUS
I1
I0
INDEX/BASE REG
3 WAY ADDER
MAP MISS
LMAR COPY
EFFECTIVE MAR
DB BUS
BYTE CONSTANT
LITERAL PROM
SIGN EXTEND
SHIFT SIGN EXT
2901
ALU A B
2901 CREG
CROM/CREG ORDERS IE - UNIT
TURNKEY PANEL
MICRO-INTERRUPT CONTROL
2:1 MUX
N COUNTER
CACHE DATA OUT REG
NIBBLE SHIFTER
I3
I2
CROM
MUX
SCRATCH PAD
LEADING TRAILING ZEROS
AUTO IPL
MDB BUS
RAM DATA IN
RAM DATA OUT
µPC SAVE
6:1 MUX
µPC
CONTROL STORE (CROM)
CROM BUS
CROM/CREG ORDERS MS - UNIT
830484

Figure 3-2. 32/67 CPU Block Diagram

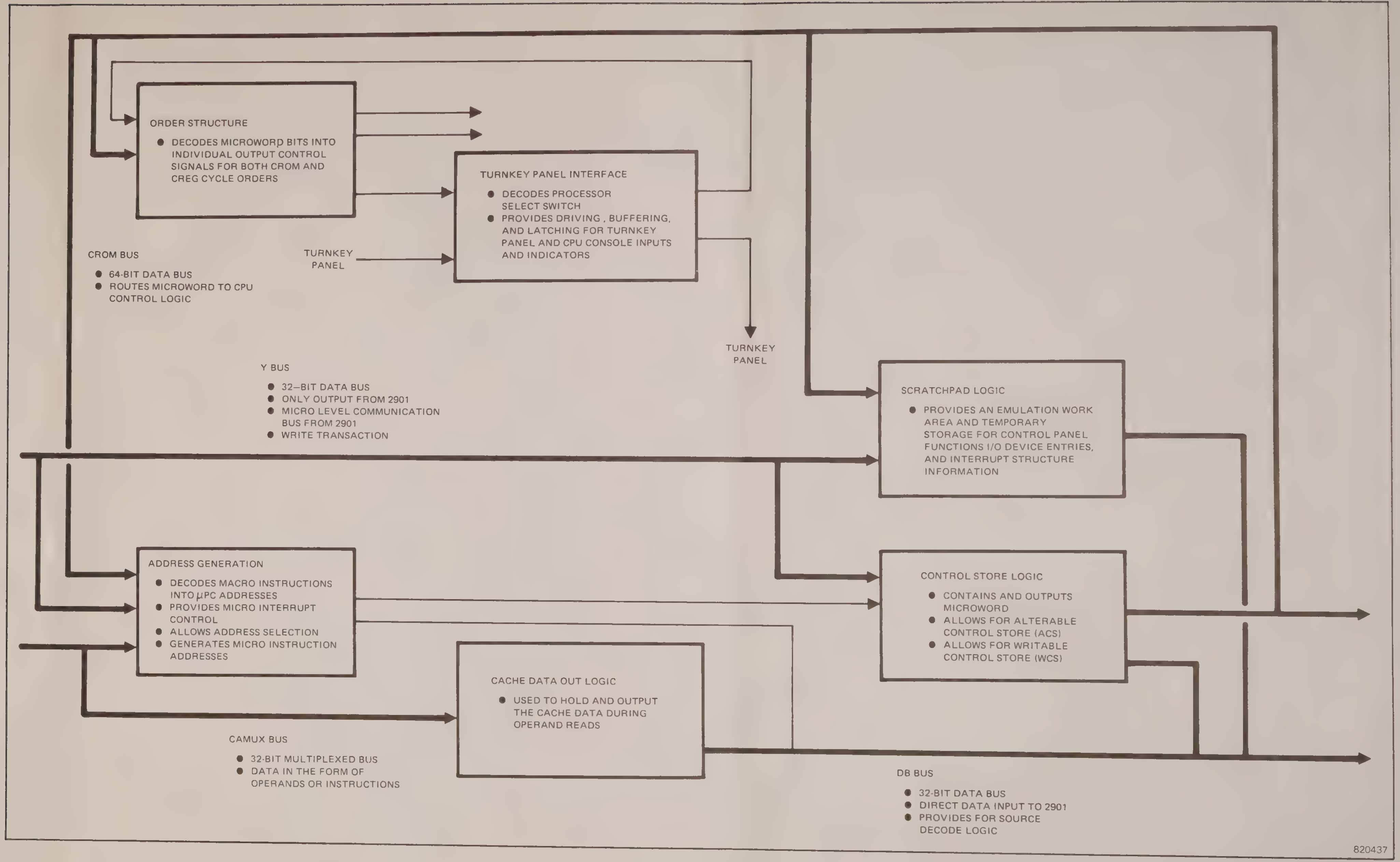

ORDER STRUCTURE
● DECODES MICROWORD BITS INTO INDIVIDUAL OUTPUT CONTROL SIGNALS FOR BOTH CROM AND CREG CYCLE ORDERS
TURNKEY PANEL INTERFACE
● DECODES PROCESSOR SELECT SWITCH
● PROVIDES DRIVING, BUFFERING, AND LATCHING FOR TURNKEY PANEL AND CPU CONSOLE INPUTS AND INDICATORS
CROM BUS
● 64-BIT DATA BUS
● ROUTES MICROWORD TO CPU CONTROL LOGIC
TURNKEY PANEL
TURNKEY PANEL
Y BUS
● 32–BIT DATA BUS
● ONLY OUTPUT FROM 2901
● MICRO LEVEL COMMUNICATION BUS FROM 2901
● WRITE TRANSACTION
SCRATCHPAD LOGIC
● PROVIDES AN EMULATION WORK AREA AND TEMPORARY STORAGE FOR CONTROL PANEL FUNCTIONS I/O DEVICE ENTRIES, AND INTERRUPT STRUCTURE INFORMATION
ADDRESS GENERATION
● DECODES MACRO INSTRUCTIONS INTO μPC ADDRESSES
● PROVIDES MICRO INTERRUPT CONTROL
● ALLOWS ADDRESS SELECTION
● GENERATES MICRO INSTRUCTION ADDRESSES
CONTROL STORE LOGIC
● CONTAINS AND OUTPUTS MICROWORD
● ALLOWS FOR ALTERABLE CONTROL STORE (ACS)
● ALLOWS FOR WRITABLE CONTROL STORE (WCS)
CACHE DATA OUT LOGIC
● USED TO HOLD AND OUTPUT THE CACHE DATA DURING OPERAND READS
CAMUX BUS
● 32-BIT MULTIPLEXED BUS
● DATA IN THE FORM OF OPERANDS OR INSTRUCTIONS
DB BUS
● 32-BIT DATA BUS
● DIRECT DATA INPUT TO 2901
● PROVIDES FOR SOURCE DECODE LOGIC
820437

Figure 3-3. MS Unit Functional Diagram

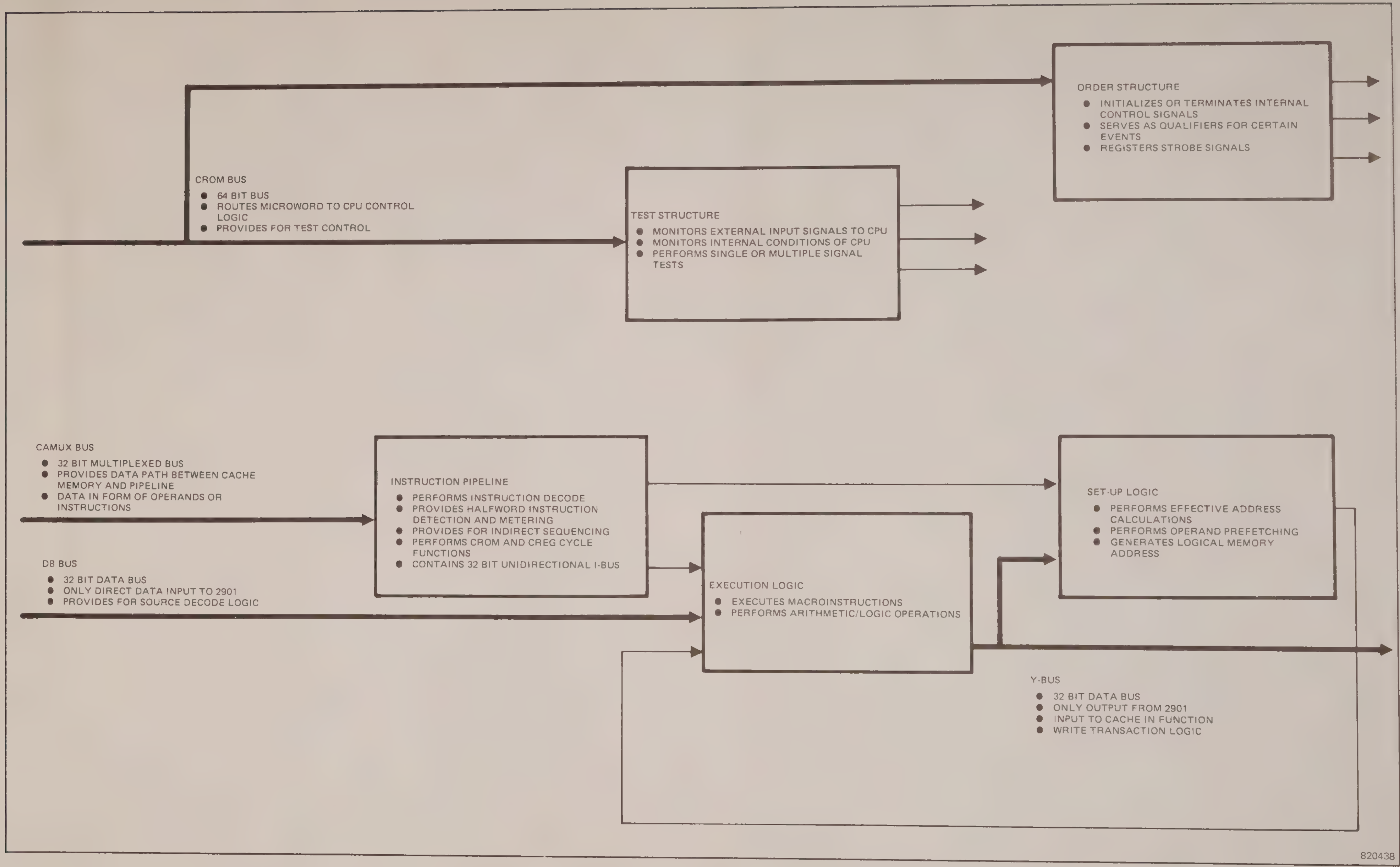

ORDER STRUCTURE
● INITIALIZES OR TERMINATES INTERNAL CONTROL SIGNALS
● SERVES AS QUALIFIERS FOR CERTAIN EVENTS
● REGISTERS STROBE SIGNALS
CROM BUS
● 64 BIT BUS
● ROUTES MICROWORD TO CPU CONTROL LOGIC
● PROVIDES FOR TEST CONTROL
TEST STRUCTURE
● MONITORS EXTERNAL INPUT SIGNALS TO CPU
● MONITORS INTERNAL CONDITIONS OF CPU
● PERFORMS SINGLE OR MULTIPLE SIGNAL TESTS
CAMUX BUS
● 32 BIT MULTIPLEXED BUS
● PROVIDES DATA PATH BETWEEN CACHE MEMORY AND PIPELINE
● DATA IN FORM OF OPERANDS OR INSTRUCTIONS
INSTRUCTION PIPELINE
● PERFORMS INSTRUCTION DECODE
● PROVIDES HALFWORD INSTRUCTION DETECTION AND METERING
● PROVIDES FOR INDIRECT SEQUENCING
● PERFORMS CROM AND CREG CYCLE FUNCTIONS
● CONTAINS 32 BIT UNIDIRECTIONAL I-BUS
SET-UP LOGIC
● PERFORMS EFFECTIVE ADDRESS CALCULATIONS
● PERFORMS OPERAND PREFETCHING
● GENERATES LOGICAL MEMORY ADDRESS
DB BUS
● 32 BIT DATA BUS
● ONLY DIRECT DATA INPUT TO 2901
● PROVIDES FOR SOURCE DECODE LOGIC
EXECUTION LOGIC
● EXECUTES MACROINSTRUCTIONS
● PERFORMS ARITHMETIC/LOGIC OPERATIONS
Y-BUS
● 32 BIT DATA BUS
● ONLY OUTPUT FROM 2901
● INPUT TO CACHE IN FUNCTION
● WRITE TRANSACTION LOGIC
820438

Figure 3-4. IE Unit Functional Diagram

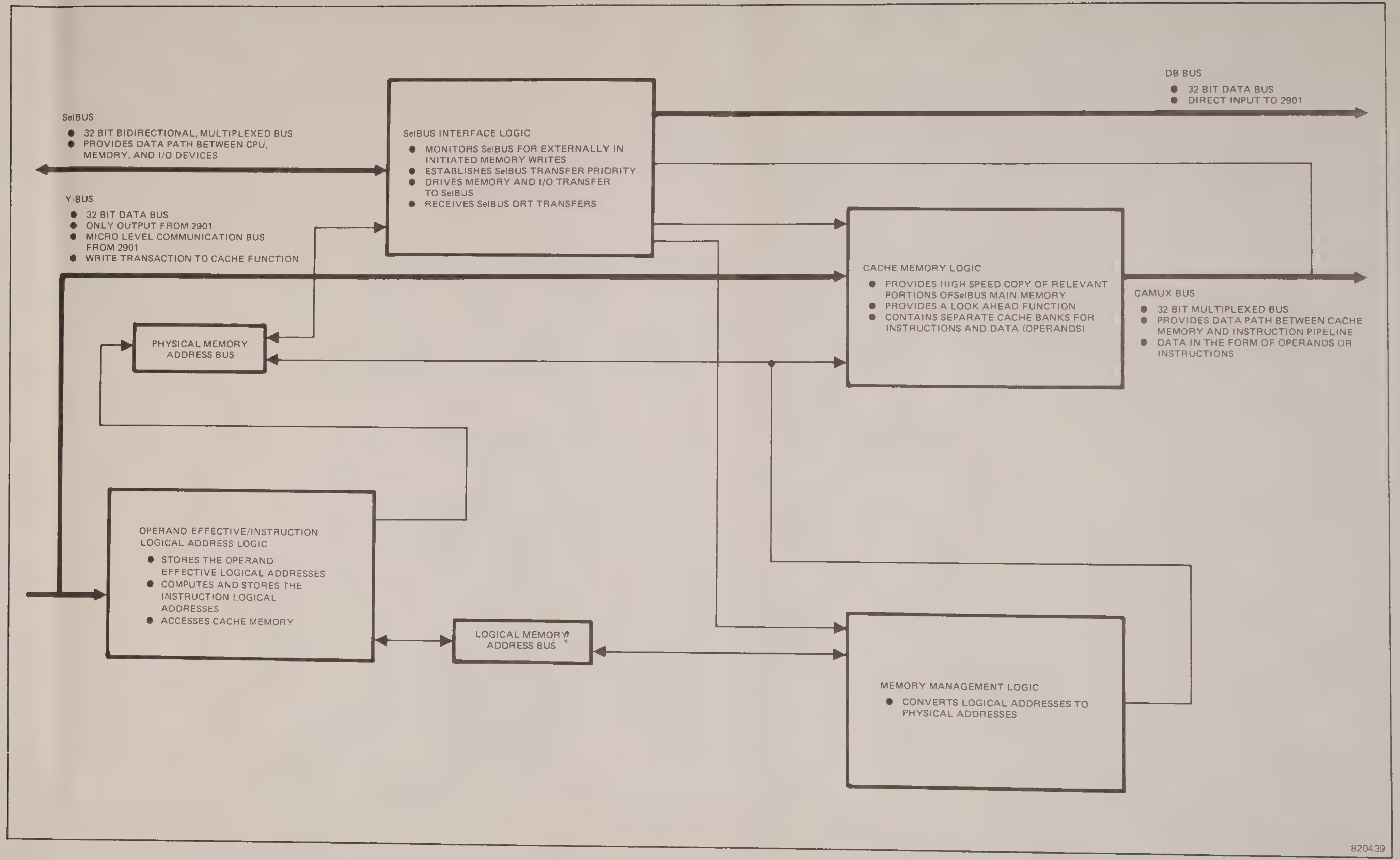

SelBUS
32 BIT BIDIRECTIONAL, MULTIPLEXED BUS
PROVIDES DATA PATH BETWEEN CPU, MEMORY, AND I/O DEVICES
Y-BUS
32 BIT DATA BUS
ONLY OUTPUT FROM 2901
MICRO LEVEL COMMUNICATION BUS FROM 2901
WRITE TRANSACTION TO CACHE FUNCTION
SelBUS INTERFACE LOGIC
MONITORS SelBUS FOR EXTERNALLY IN INITIATED MEMORY WRITES
ESTABLISHES SelBUS TRANSFER PRIORITY
DRIVES MEMORY AND I/O TRANSFER TO SelBUS
RECEIVES SelBUS DRT TRANSFERS
DB BUS
32 BIT DATA BUS
DIRECT INPUT TO 2901
CACHE MEMORY LOGIC
PROVIDES HIGH SPEED COPY OF RELEVANT PORTIONS OFSelBUS MAIN MEMORY
PROVIDES A LOOK AHEAD FUNCTION
CONTAINS SEPARATE CACHE BANKS FOR INSTRUCTIONS AND DATA (OPERANDS)
CAMUX BUS
32 BIT MULTIPLEXED BUS
PROVIDES DATA PATH BETWEEN CACHE MEMORY AND INSTRUCTION PIPELINE
DATA IN THE FORM OF OPERANDS OR INSTRUCTIONS
PHYSICAL MEMORY ADDRESS BUS
OPERAND EFFECTIVE/INSTRUCTION LOGICAL ADDRESS LOGIC
STORES THE OPERAND EFFECTIVE LOGICAL ADDRESSES
COMPUTES AND STORES THE INSTRUCTION LOGICAL ADDRESSES
ACCESSES CACHE MEMORY
LOGICAL MEMORY ADDRESS BUS
MEMORY MANAGEMENT LOGIC
CONVERTS LOGICAL ADDRESSES TO PHYSICAL ADDRESSES
820439

Figure 3-5. CS Unit Functional Diagram

3.2.3.2 Operand Effective/Instruction Logical Address Logic

Functions

1) Stores the operand effective logical addresses

2) Computes and stores the instruction logical addresses

3.2.3.3 Memory Management Logic

Functions

1) Converts logical addresses to physical addresses

2) Performs mapping functions

3.2.3.4 Cache Memory Logic

Functions

1) Provides a high-speed copy of standard SelBUS memory

2) Provides a look-ahead to enhance CPU performance

3.2.3.5 Machine Cycles

Each microinstruction uses two time periods for execution: the CROM cycle and the CREG (control register) cycle (see figure 4-2). The timing that regulates these cycles is a single-phase clock that triggers every 150 nanoseconds throughout the system.

The CROM cycle refers to functions performed at the end of the control store access cycle. These functions are performed by direct output or decodes of the CROM without an intervening clock. Typically these include microsequencing and operation monitoring (test) functions.

The CREG cycle refers to functions performed from the control store register. CREG cycle functions require an intervening clock between them and CROM cycle functions and can provide a result-oriented operation for a full step (150 nsec) following the CROM step. Generally, CREG cycle functions control the data structure and data put-aways.

3.2.3.6 Clock Generation

The system clock is routed to the MS Unit, IE Unit, and CS Unit as signal LCLKL. (This is a SelBUS clock. Although the prefix L is used, the active edge is positive going.) The clock signals within each unit are derived from this sytem clock input. The stop system clock signal LSTSC is also routed to each unit. LSTSC stops all clocks with the exception of L2FREERUNCLK, which is used in ACS and WCS RAM control. Figure 3-6 is a composite illustration showing basic timing, three clock levels, active edges denoted by arrows, and clock widths. The convention used for clock signal nomenclature describes each signal; an example is included in the illustration. The CPU stopclock HSTOPCLK controls all clock signals having the suffix S (stopclocks).

3.3 Microsequencer (MS) Unit

The microsequencer unit contains the CPU microprogram in PROM; the PROM array is referred to as the control read only memory (CROM). The MS unit has facilities for ACS (alterable control store) and WCS (writable control store) functions.

The CPU microprogram counter (uPC) is located on the MS PCB. The order structure within the MS Unit generates (from the CROM microword) CROM orders, CREG (control register) orders and latch orders that control the operation of the MS unit and CPU. Operations such as macroinstruction decode, microinterrupt control, uPC increment, uPC jump, or uPC branch are processed within the MS unit.

The following text is presented in two major subdivisions: Microsequencer Components and Microsequencer Operation.

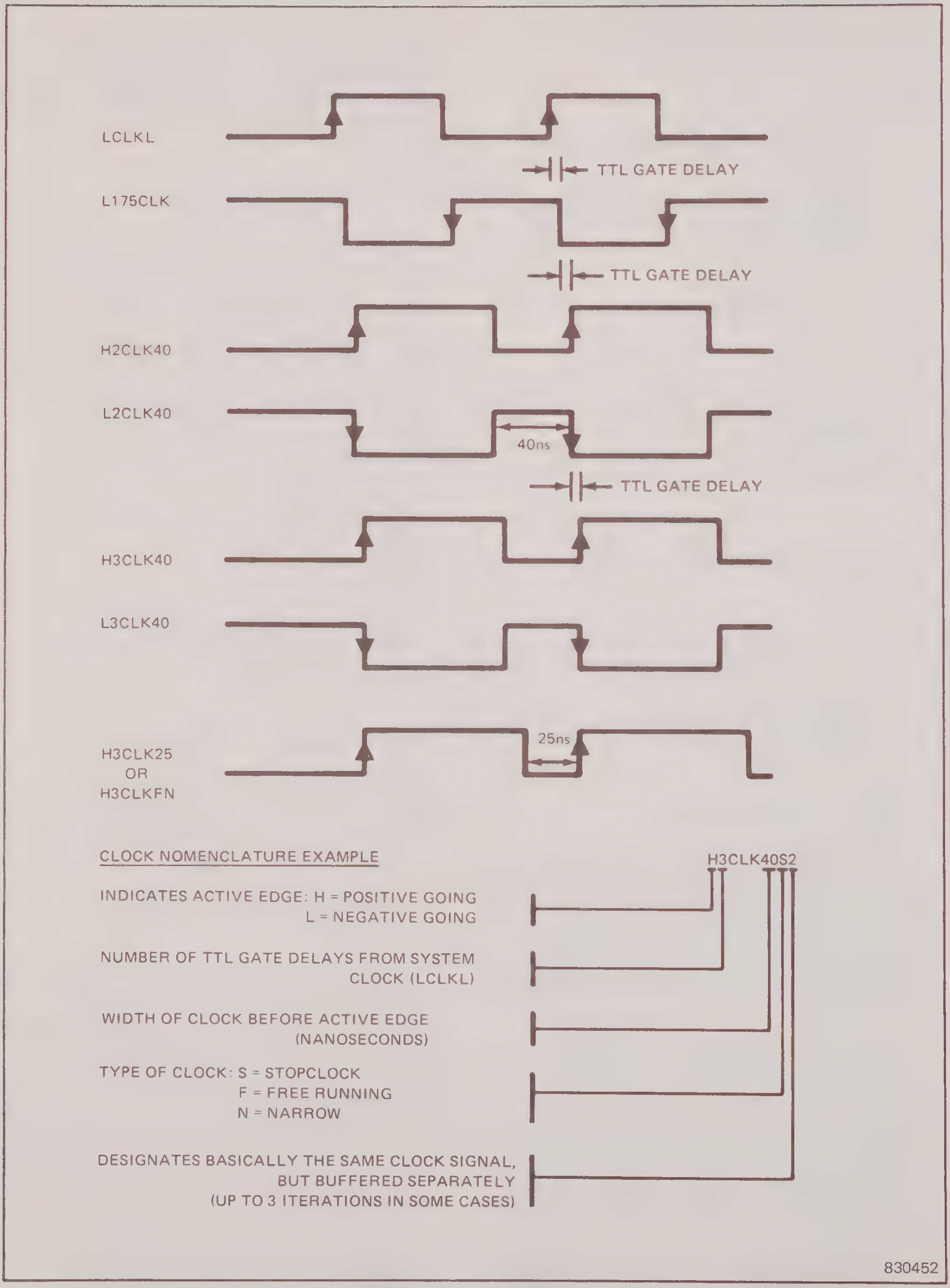

LCLKL
TTL GATE DELAY
L175CLK
TTL GATE DELAY
H2CLK40
L2CLK40
40ns
TTL GATE DELAY
H3CLK40
L3CLK40
H3CLK25 OR H3CLKFN
25ns
CLOCK NOMENCLATURE EXAMPLE
H3CLK40S2
INDICATES ACTIVE EDGE: H = POSITIVE GOING
L = NEGATIVE GOING
NUMBER OF TTL GATE DELAYS FROM SYSTEM CLOCK (LCLKL)
WIDTH OF CLOCK BEFORE ACTIVE EDGE (NANOSECONDS)
TYPE OF CLOCK: S = STOPCLOCK
F = FREE RUNNING
N = NARROW
DESIGNATES BASICALLY THE SAME CLOCK SIGNAL, BUT BUFFERED SEPARATELY (UP TO 3 ITERATIONS IN SOME CASES)
830452

Figure 3-6. Basic Timing Diagram and Clock Nomenclature

3.3.1 Microsequencer Components (see figure 3-7)

Note

Items in parentheses (xx) denote logic drawing 130-103568 page numbers. These are included as supplemental information for Gould S.E.L. field service personnel. Where applicable, these numbers are also included on block diagrams.

3.3.1.1 Y Bus and DB Bus

The 32-bit Y Bus is the only output from the MP2901 (IE Unit). The data is distributed to the base/index registers (IE Unit) and, after being buffered (YB Bus), is routed to the CS Unit and MS Unit.

The 32-bit DB Bus provides the only direct data input to the MP2901. Inputs to the DB Bus are provided by the effective memory address register (IE Unit), the I0 register (IE Unit), or the RAM data out register (MS Unit). On the MS unit, this bus originates as the MD Bus. After buffering on the MS Unit, it is termed the MDB Bus. It is connected to the DB Bus at the IE Unit.

3.3.1.2 Right Shifter (34)

The right shifter is 32-bits wide and is used during cache reads to align data to byte, halfword, or word boundaries. Input data to the right shifter is provided by the CAMUX Bus. The output of the right shifter is gated to the cache data out even and cache data out odd registers if it is an operand read. CAMUX is gated to I3 register if it is an instruction read, and the I2 indirect register (in the IE unit) if it is an indirect read. The right shifter is designed to shift either 1, 2, or 3 bytes to the right.

3.3.1.3 Cache Data Out Registers (4,35)

Two 32-bit cache data out registers are provided to receive operands read from cache. The registers are the cache data out even and the cache data out odd. During all operand cache accesses (except doubleword pairs) the even register always receives the even or odd cache output. When the effective address specifies a doubleword pair, the even register receives cache outputs from the even addressed cache words and the odd register receives cache outputs from the odd addressed cache words. The output of the cache data out registers is gated to the MDB Bus. The output from the cache data out even register may be gated via the MDB Bus to the sign extend logic (sheet 5).

3.3.1.4 I3 and I2 MS Copy Registers (27)

The two I3 MS copy registers are 16-bit registers that maintain a copy of the right and left half of I3 register in the IE Unit (the top level of the instruction pipeline). The I3 MS copy register is loaded with the cache output during instruction cache read operations. The output of only one I3 register is enabled and gated to the I2 MS copy register.

3.3.1.5 Instruction Decode (28,29)

The instruction decode (vector decode logic) consists of a decode array driven by the I2 MS copy register (I2 Bus copy) and decodes macroinstruction op-code, augment code, and sub-op-code fields into uPC addresses. In addition to the op-code fields the instruction decode also checks the right hand flag, base mode flag, and FPA present flag in determining the uPC address.

The instruction decode (vector decode logic) relationship between op-codes, modes (flags), and uPC addresses are predefined and programmed (burned) into the decode array. The output of the instruction decode (the uPC address) is gated to the 6:1 input address selection multiplexer. The instruction decode (decode array) can be considered a look-up table for op-codes and flags (inputs) versus uPC addresses (outputs). An instruction in the instruction decode phase is, at a minimum, two clocks (300ns) away from execution.

3.3.1.6 Decode Save (29)

The decode save register saves the uPC address output of the instruction decode. In the event of microinterrupt, the uPC control may select the output of the decode save (through the 6:1 mux) to serve as a microinterrupt return address. The output of the decode save register is only used if a valid instruction is in the I1 pipeline register during a microinterrupt return sequence.

3.3.1.7 Microinterrupt Control (31)

The microinterrupt control hardware provides a means of interrupting the firmware out of macroinstruction emulation sequences. The microinterrupt must be specifically enabled by the firmware before an interrupt can occur. A microinterrupt represents an exceptional event that must be identified and handled by firmware. The exceptional event may be an error, a software interrupt or trap, or may indicate that some hardware element or interlock requires servicing. The microinterrupt control hardware provides a vectorized and prioritized decode of the conditions causing the microinterrupt and generates a uPC address of a firmware routine designed to handle the exceptional event.

If the exceptional event is an error, trap, or interrupt, the firmware will not attempt to return to the sequence in which the microinterrupt occurred. In the event of error, trap, or interrupt, the firmware will activate a firmware trap or interrupt handler that will in turn activate a software trap or interrupt handler.

If the exceptional event represents a hardware element that requires servicing, the firmware will service that element and resume processing at the point at which the microinterrupt occurred. To accommodate the microinterrupt the hardware provides a uPC save register to store the interrupted context. The uPC save register provides a microinterrupt return address that must be gated back through the 6:1 input address selection multiplexer to the uPC at the end of the microinterrupt routine. The microinterrupt control hardware does not provide for nested microinterrupts.

3.3.1.8 6:1 Input Address Selection Multiplexer (24,25)

The input address selection multiplexer (6:1 mux) selects the source of the next uPC address. The address selection provides uPC branch, jump, and return capabilities. The selectable address sources are:

1. Microword (CROM)
2. Decode Save
3. Instruction Decode
4. IE Unit data structure (YB-Bus)
5. Microinterrupt vector from IE unit
6. uPC Save Register

- Microword (CROM)

  The uPC control directs the 6:1 mux to select either 4, 8, or 12 bits of CROM bits 52 through 63 to the uPC during Hop (Seq0), Leap (Seq1), Branch (Seq2), operations.

- Decode save (refer to paragraph 3.3.1.6).

- Instruction decode (refer to paragraph 3.3.1.5).

- IE unit data structure (YB Bus).

- The uPC control directs the 6:1 mux to select the YB bus input to the uPC during branch (Seq3) and jump data orders from CROM.

- Microinterrupt vector. When a microinterrupt occurs, the microinterrupt vector generated by the IE unit is chosen by the uPC and vectors to a firmware routine where the microinterrupt is serviced.

- Microprogram counter save register (refer to paragraph 3.3.1.10).

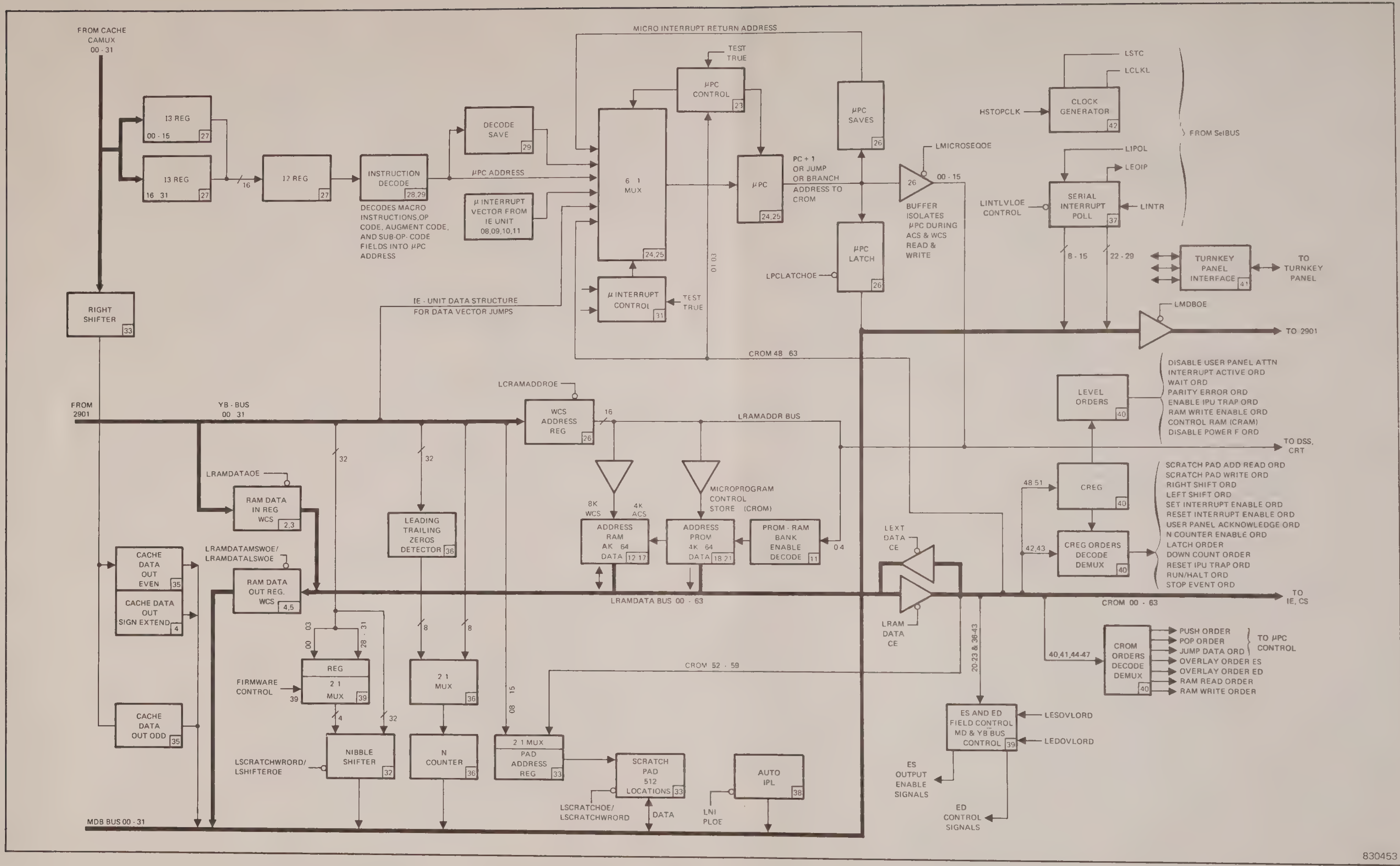

FROM CACHE CAMUX 00 - 31
I3 REG 00 - 15 27
I3 REG 16 31 27
16
I2 REG 27
INSTRUCTION DECODE 28,29
DECODES MACRO INSTRUCTIONS,OP CODE, AUGMENT CODE, AND SUB-OP- CODE FIELDS INTO μPC ADDRESS
DECODE SAVE 29
μPC ADDRESS
μ INTERRUPT VECTOR FROM IE UNIT 08,09,10,11
MICRO INTERRUPT RETURN ADDRESS
TEST TRUE
μPC CONTROL 23
6 1 MUX 24,25
μ INTERRUPT CONTROL 31
TEST TRUE
01 03
μPC 24,25
PC + 1 OR JUMP OR BRANCH ADDRESS TO CROM
μPC SAVES 26
μPC LATCH 26
LPCLATCHOE
LMICROSEQOE
26
00 - 15
BUFFER ISOLATES μPC DURING ACS & WCS READ & WRITE
LSTC
LCLKL
HSTOPCLK
CLOCK GENERATOR 42
FROM SelBUS
L1POL
LEOIP
LINTLVLOE CONTROL
SERIAL INTERRUPT POLL 37
LINTR
8 - 15
22 - 29
TURNKEY PANEL INTERFACE 41
TO TURNKEY PANEL
LMDBOE
TO 2901
RIGHT SHIFTER 33
IE - UNIT DATA STRUCTURE FOR DATA VECTOR JUMPS
CROM 48 63
LCRAMADDROE
FROM 2901
YB - BUS 00 31
WCS ADDRESS REG 26
16
LRAMADDR BUS
LEVEL ORDERS 40
DISABLE USER PANEL ATTN
INTERRUPT ACTIVE ORD
WAIT ORD
PARITY ERROR ORD
ENABLE IPU TRAP ORD
RAM WRITE ENABLE ORD
CONTROL RAM (CRAM)
DISABLE POWER F ORD
TO DSS, CRT
32
32
LRAMDATAOE
RAM DATA IN REG WCS 2,3
LEADING TRAILING ZEROS DETECTOR 36
8K WCS
4K ACS
MICROPROGRAM CONTROL STORE (CROM)
ADDRESS RAM AK 64 DATA 12 17
ADDRESS PROM 4K 64 DATA 18 21
PROM - RAM BANK ENABLE DECODE 11
04
48 51
CREG 40
42,43
CREG ORDERS DECODE DEMUX 40
SCRATCH PAD ADD READ ORD
SCRATCH PAD WRITE ORD
RIGHT SHIFT ORD
LEFT SHIFT ORD
SET INTERRUPT ENABLE ORD
RESET INTERRUPT ENABLE ORD
USER PANEL ACKNOWLEDGE ORD
N COUNTER ENABLE ORD
LATCH ORDER
DOWN COUNT ORDER
RESET IPU TRAP ORD
RUN/HALT ORD
STOP EVENT ORD
LEXT DATA CE
LRAM DATA CE
CACHE DATA OUT EVEN 35
LRAMDATAMSWOE/ LRAMDATALSWOE
RAM DATA OUT REG. WCS 4,5
CACHE DATA OUT SIGN EXTEND 4
LRAMDATA BUS 00 - 63
CROM 00 - 63
TO IE, CS
03
00
28 - 31
8
8
REG
2 1 MUX 39
FIRMWARE CONTROL
39
2 1 MUX 36
CROM 52 - 59
08 15
20-23 & 36-43
40,41,44-47
CROM ORDERS DECODE DEMUX 40
PUSH ORDER
POP ORDER
JUMP DATA ORD
TO μPC CONTROL
OVERLAY ORDER ES
OVERLAY ORDER ED
RAM READ ORDER
RAM WRITE ORDER
CACHE DATA OUT ODD 35
4
32
NIBBLE SHIFTER 32
LSCRATCHWRORD/ LSHIFTEROE
N COUNTER 36
2 1 MUX PAD ADDRESS REG 33
SCRATCH PAD 512 LOCATIONS 33
LSCRATCHOE/ LSCRATCHWRORD
DATA
AUTO IPL 38
LNI PLOE
ES AND ED FIELD CONTROL MD & YB BUS CONTROL 39
LESOVLORD
LEDOVLORD
ES OUTPUT ENABLE SIGNALS
ED CONTROL SIGNALS
MDB BUS 00 - 31
830453

Figure 3-7. MS Unit Block Diagram

3.3.1.9 Microprogram Counter (uPC) (24,25)

The microprogram counter consists of a 16-bit wide adder, push/pop stack, 4:1 multiplexer, and holding register. The output of the uPC is the next microinstruction address to the control read only memory (CROM). The uPC output may also be sent to the uPC latch and/or gated into the uPC save register.

3.3.1.10 Microprogram Counter Save Register (26)

The uPC save register is used to hold the uPC address of a microinterrupted microinstruction. The contents of the save register must be gated back through the 6:1 input address selection multiplexer (6:1 Mux) to the uPC (microreturn).

3.3.1.11 Microprogram Counter Latch (26)

The uPC latch may be used to capture the output of the uPC and gate the contents to the MDB Bus to allow the firmware to perform uPC relative operations. The uPC latch is not being utilized within the CPU at this time.

3.3.1.12 Microprogram Counter Output Buffer (26)

The uPC output buffers isolate the uPC during ACS and WCS read and write operations.

3.3.1.13 WCS RAM Data Registers (2,3,4,5)

The RAM data in register transfers data from the 32-bit YB bus to the 64-bit LRAMDATA (CROM) bus. The RAM data in register is a 64-bit register which requires two operations to load. This register feeds the 64-bit LRAMDATA bus. The ED overlay field of the microword determines whether the output of the YB Bus is loaded into the MSW or LSW portion of the RAM data in register.

The RAM data out register transfers data from the 64-bit LRAMDATA (CROM) bus to the 32-bit MD bus. The RAM data out register is a 64-bit register and requires two operations to gate data to the 32-bit MD bus. The ES overlay field of the microword determines whether the MSW or LSW of the microword (CROM bus) will be gated to the MD bus.

3.3.1.14 WCS Address Register (12)

The WCS address register is loaded with bits 16 through 31 of the YB Bus and gated to the HRAMADDR lines 00 through 15. The WCS address comes from one of the GPRs within the MP2901 specified by the RWCS or WWCS instructions. The register is used during a read ACS and WCS or write ACS and WCS.

3.3.1.15 N-Counter (36)

The N-counter is used by the CPU firmware to supply iteration count for repetative operations such as shift, multiply, and divide. The N-counter provides a count for an address designation in the performance of a jump or branch. The counter may be loaded from the leading/trailing zeros detect logic or from the YB Bus. The ouput may be gated to the MD Bus.

All flip-flops in each counter simultaneously clock so that the outputs change coincident with each other when instructed by the count enable order (LCNTENORD) and internal gating. When the down count order (LDNCNTORD) goes low, the counter counts down; and when high, the counter counts up. Input to the N-counter is controlled by a 2:1 mux. The 2:1 mux selects either the YB Bus (00 through 07), or the YB Bus via the leading trailing zeros detector. The N counter can count up to 256. The N counter overflow signal (LNOVRFLOW) sets a flag in the status port that the N counter contains all ones in the count up mode and all zeros in the count down mode.

3.3.1.16 Control Store RAM (12-17)

The control store RAM is a 12K by 64-bit RAM array that is divided into 8K of WCS RAM and 4K of ACS RAM. The WCS RAM locations are considered add-on control store for the control read only memory (microprogram control store) located in PROM. The ACS is a copy (shadow) of the PROM control store, may be written into under software control, and may be the source of the primary execution microcode when the CPU is in the operational state. Data to and from the RAM is by way of the LRAMDATA Bus (00 through 63). ACS locations can only be accessed by software WCS instructions.

3.3.1.17 Control Store PROM (18-21)

The control store PROM is a 4K by 64-bit control read only memory (CROM) that contains the CPU microprogram. Data output from the address PROM is gated to the LRAMDATA Bus (00 through 63). The output of the PROM is a microword that is to be decoded to control the CPU. Software can read the contents of the PROM by means of the RWCS instruction.

3.3.1.18 PROM & RAM Bank Enable Decode (11)

The PROM and RAM bank enable decode determines which address range the incoming address falls under by decoding address lines 00 through 04 and generates the corresponding bank output enable to enable either the ACS/WCS RAM or the PROM.

3.3.1.19 Order Structure (40)

The order structure is an output signal array that decodes microword bits into individual output control signals. From one to four control signals (orders) may be generated by a single microword. The order structure provides both CROM and CREG cycle orders.

In general, orders are used to initialize or terminate internal control functions. Orders may also serve as qualifiers for certain events and as register strobe signals. Normally orders are 150 ns pulses; however, the CPU also provides 16 level-orders that may be ordered to a set-or-reset state by the firmware.

3.3.1.20 Scratchpad (33)

The scratch pad logic consists of eight 1024 x 4 RAMs that gate data to and from the MDB Bus. The scratch pad address is selected by the scratch pad address selection mux that can select either the microword (bits 52 through 59) or the YB Bus (bits 08 through 15) as the address source. The scratch pad is used to map software I/O channels, I/O device addresses, and software interrupt levels into SelBUS physical addresses and interrupt levels. Sixteen locations are reserved for software apparent parameters such as interrupt and trap table base addresses. Only 512 locations of the 1K RAMs are used to store scratch pad information. The scratch pad is loaded by the 2901 through the nibble shifter. Locations 100 through 1FF are reserved for firmware and are not accessible by software.

3.3.1.21 Nibble Shifter (32)

The MS Unit nibble shifter is 32-bits wide and has the capability of left or right 4-bit shifts. Right and left shift orders are CREG orders. The LS.MS. FILL is an order that tells the nibble shifter to choose either the LS FILL (bits 28 through 31) or the MS FILL (bits 0 through 3). The shifter may be zero filled or the result of the last shift may be used to provide a two-pass double-precision shift. The capability of circular shift left or right also exists.

3.3.1.22 Auto IPL (38)

Auto IPL provides the CPU with the capability of initiating an initial program load (IPL) program (software bootstrap program) during power-up initialization with memory contents that do not meet the requirements of the power-up automatic restart (auto-restart) feature.

Sixteen auto-IPL jumpers are used to enable the auto-IPL feature and provide a 15-bit SelBUS physical address and sub-address of a dedicated I/O device that contains the software bootstrap program. Typically, this I/O device is the system disc containing the operating system software.

3.3.1.23 Serial Interrupt Poll (37)

The duration of the interrupt polling cycle is controlled by a binary counter that counts nine clocks and resets to zero. The interrupt level that wins the poll is fed to an 8-bit serial to parallel shift register and the contents gated to two parallel registers that output to the MDB Bus, bits 08 through 15 and bits 22 through 29. On the 9th clock after the interrupt poll is initiated, the parallel registers are gated to the MDB Bus. The output of the serial interrupt poll is a ones complement of the winning interrupt level.

The interrupt EOIP signal is generated only by the CPU. The IPU samples this line to synchronize its interrupt logic.

3.3.1.24 ES and ED Field Control (39)

The ES field is derived from CROM bits 20 through 23 (which are the inputs to three decode demultiplexers) and controls gating to the MDB Bus. The ES field overlay is initiated by LESOVLORD. The decoded outputs are latched before enabling the DB bus. The latching method assures proper timing of these signals so that all coordinating registers receive their enabling signals within a nominal margin.

The ED field overlay is initiated by CREG clock and LEDOVLORD. It is used specify the destination of the MP2901 Y Bus. Three CREG bits are generated (CREG 41, 42, and 43) to enable other groups within the order structure.

3.3.1.25 Turnkey Panel Interface (40)

The turnkey panel interface contains logic to decode the PROCESSOR SELECT switch on the turnkey panel (CPU/IPU). The interface contains drivers, buffers, and latches for indicators on the turnkey panel and the CPU console.

3.3.2 Microsequencer Operation

3.3.2.1 Microprogram Counter (see figure 3-8)

The uPC consist of four 4-bit slice elements that provide a 16-bit data path. Each element is comprised of an output register, 4:1 mux, push-pop stack, and full adder.

- The output register provides a steady-state output throughout each clock cycle.
- Each 4:1 input source select multiplexer selects, independently, through the use of control lines S5 and S6, a 4-bit segment of one or more of the four input sources for the assembly of the next uPC address.
- The push-pop stack is used for nesting up to four levels of sub-routine return addresses. In the LOAD mode, the top (first) word is loaded with data from the full adder and any previous data at that location is lost, all other locations in the stack remain unchanged. In the PUSH mode, the top word is loaded with data from the full adder and any previous data at that location, and data in the second and third word positions, is pushed down one word location and retained.
- Any data located in the bottom word position during the push operation is lost. In the POP mode, data (words) in the stack are moved up one position during each pop operation. Data located in the fourth (bottom) location, during successive pop operations, is retained by the bottom position and duplicated in the third, second, and finally the first (top) position, filling the stack with fourth word location data.

- The full adder is a four-function ALU; however, only one of its function is used for simple incrementation of input to port B by one by clamping S1 and C-in to a positive voltage and grounding S2. This function is used when the sequence field specifies a NOP, or when HTESTTRUE is low indicating test false.

3.3.2.2 Microprogram Counter Control

The uPC control receives bits 01, 02, and 03 from the CROM (microword field S), high test true from the test structure, CROM orders Push, Pop and Jump Data; and a signal from microinterrupt control (interrupt branch decode latch). Output from the uPC control determines the source of the next uPC address to access the RAMs by controlling the 6:1 address selection multiplexer and the microsequencer.

3.3.2.3 Microinterrupt Control

The microinterrupt control receives high test true, microreturn, and decode signals from the uPC control, and branch taken, microinterrupt, and flush pipeline signals from the IE Unit. Outputs from the microinterrupt control include SELMICRINT and level 3 clocks that control the 6:1 mux. The 6:1 mux is a combination of a 4:1 mux and a 2:1 mux. The 4:1 mux is controlled by PCSELA, PCSELB from the uPC control and disabled by the true state of HSELMICROINT from the microinterrupt control. The 2:1 mux is controlled by HMICROINTDL and LSELMICROINT and selects either decode save or the microinterrupt bits from the IE Unit.

3.3.2.4 Serial Interrupt Poll and IPU Trap

CPU or IPU mode is determined by the LCPUMODEL signal originating in the MS Unit. This signal is used as the select control line to a multiplexer, which selects either three CPU related lines or three IPU related lines. These lines are a serial poll input, an interrupt line, and a system panel attention line.

The interrupt polling line, LIPOLL for CPU or LIPIPOLL for IPU, is driven by the device having the current highest priority. The serial interrupt level represents the priority level gated to the SelBUS. This is clocked into an eight-bit parallel-out serial shift register and then latched for enabling onto the MDB Bus. The duration of the interrupt polling cycle is controlled by a binary counter which counts nine clocks and resets to zero.

The interrupt request line, LINTR for CPU or LIPINTR for IPU, is driven low at the end of a polling sequence. The interrupt request is removed when the polling device receives an acknowledge.

The system panel attention lines, LSCPATTN for CPU or LIPATTN for IPU, indicate that a system panel type input requires servicing. The CPU or IPU initiates the required SelBUS sequence.

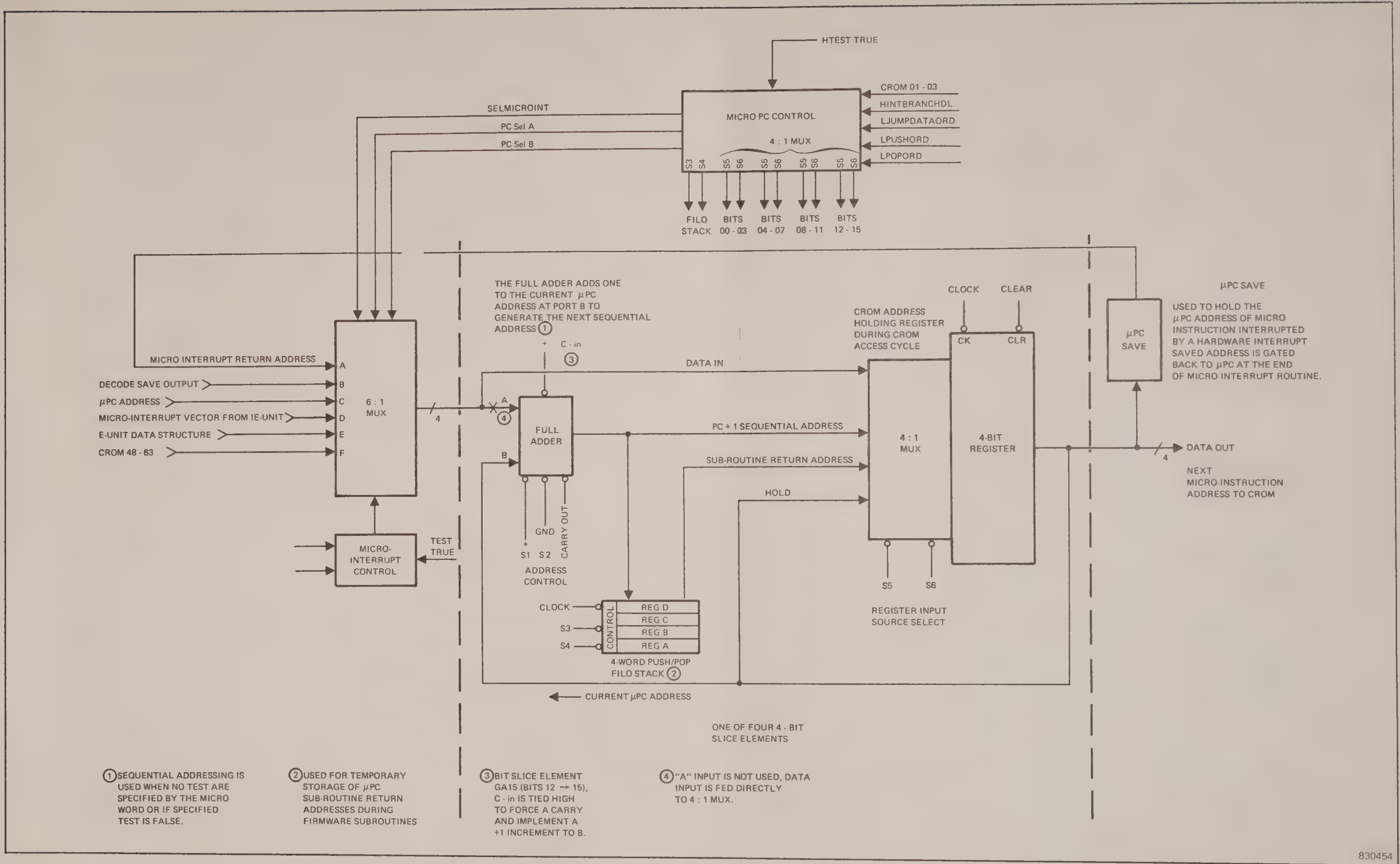
The end of interupt polling (LEOP) signal is generated only by the CPU. The IPU samples this line to synchronize its interrupt logic.

The signal CPU (LSCPU) and signal IPU (SIPU) lines are used as attention lines between the two processors. In the CPU mode, the LSIPU line is driven and the LSCPU line is sampled. In the IPU mode, the LSCPU line is driven and the LSIPU line is sampled. For either mode, a signal is returned on the driven line which activates an IPU trap flag (HIPUFF). This qualifier is routed to the interrupt logic. Two off-line signals (LCPUOFFLINE and LIPUOFFLINE) are monitored to prevent the LSCPU or LSIPU lines from being driven if the CPU or IPU is off line.

3.3.2.5 Data Flow (see figure 3-9)

3.3.2.5.1 I2 Bus

The two I3 registers maintain a copy of the I3 register in the IE Unit. The I3 copy registers are loaded with the cache output during instruction cache read operations. The output of only one I3 register is


HTEST TRUE
MICRO PC CONTROL
4 : 1 MUX
CROM 01 - 03
HINTBRANCHDL
LJUMPDATAORD
LPUSHORD
LPOPORD
S3
S4
S5
S6
S5
S6
S5
S6
S5
S6
FILO STACK
BITS 00 - 03
BITS 04 - 07
BITS 08 - 11
BITS 12 - 15
SELMICROINT
PC Sel A
PC Sel B
MICRO INTERRUPT RETURN ADDRESS
DECODE SAVE OUTPUT
µPC ADDRESS
MICRO-INTERRUPT VECTOR FROM IE-UNIT
E-UNIT DATA STRUCTURE
CROM 48 - 63
A
B
C
D
E
F
6 : 1 MUX
4
MICRO-INTERRUPT CONTROL
TEST TRUE
+
C - in
③
A
④
B
FULL ADDER
DATA IN
PC + 1 SEQUENTIAL ADDRESS
SUB-ROUTINE RETURN ADDRESS
HOLD
+
S1
GND
S2
CARRY OUT
ADDRESS CONTROL
CLOCK
S3
S4
CONTROL
REG D
REG C
REG B
REG A
4-WORD PUSH/POP FILO STACK ②
CURRENT µPC ADDRESS
CROM ADDRESS HOLDING REGISTER DURING CROM ACCESS CYCLE
CLOCK
CLEAR
CK
CLR
4 : 1 MUX
4-BIT REGISTER
S5
S6
REGISTER INPUT SOURCE SELECT
µPC SAVE
µPC SAVE
DATA OUT
4
NEXT MICRO-INSTRUCTION ADDRESS TO CROM
ONE OF FOUR 4 - BIT SLICE ELEMENTS
830454

THE FULL ADDER ADDS ONE TO THE CURRENT µPC ADDRESS AT PORT B TO GENERATE THE NEXT SEQUENTIAL ADDRESS ①

USED TO HOLD THE µPC ADDRESS OF MICRO INSTRUCTION INTERRUPTED BY A HARDWARE INTERRUPT SAVED ADDRESS IS GATED BACK TO µPC AT THE END OF MICRO INTERRUPT ROUTINE.

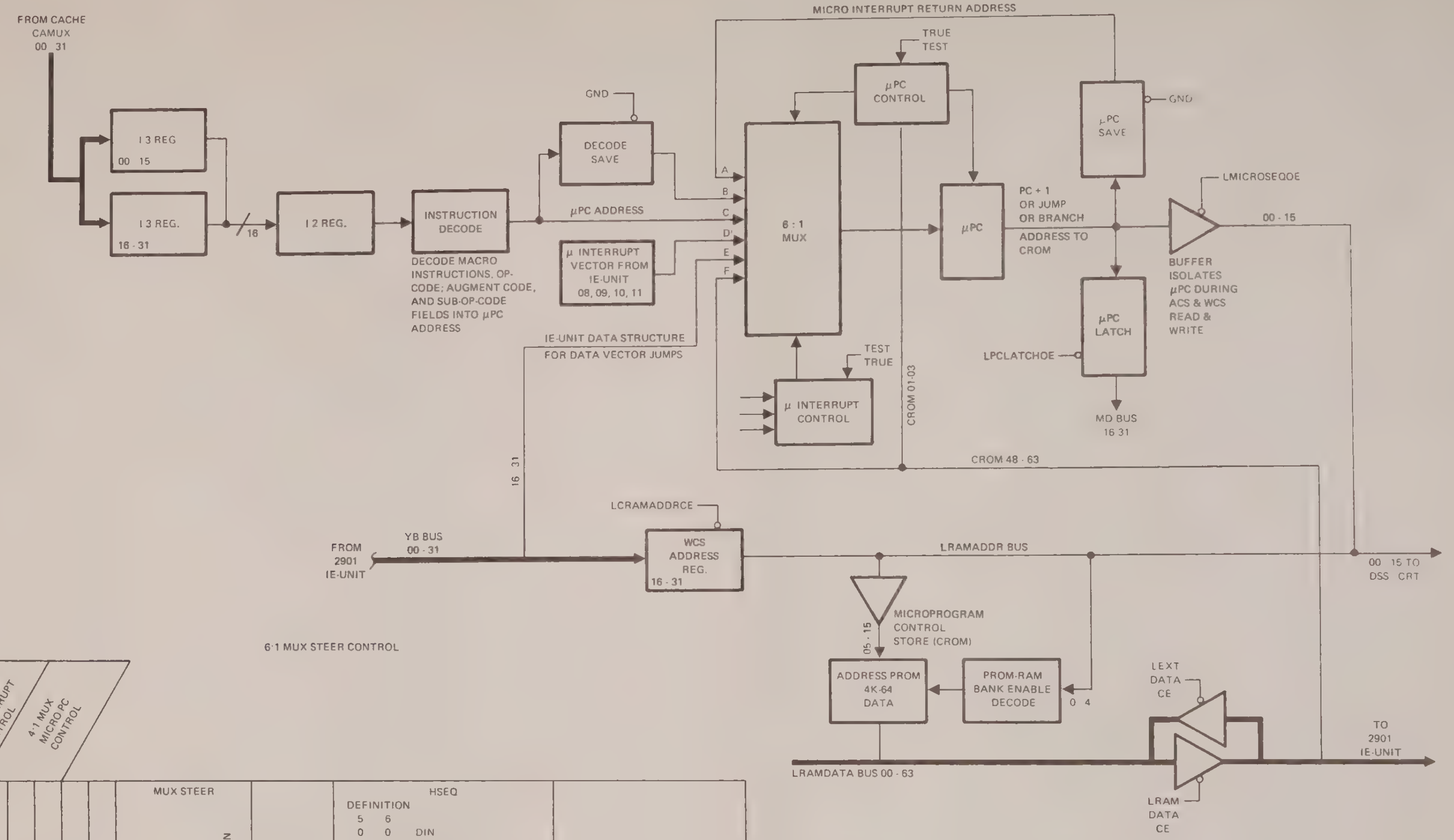
①SEQUENTIAL ADDRESSING IS USED WHEN NO TEST ARE SPECIFIED BY THE MICRO WORD OR IF SPECIFIED TEST IS FALSE.

②USED FOR TEMPORARY STORAGE OF µPC SUB-ROUTINE RETURN ADDRESSES DURING FIRMWARE SUBROUTINES

③BIT SLICE ELEMENT GA15 (BITS 12 → 15), C - in IS TIED HIGH TO FORCE A CARRY AND IMPLEMENT A +1 INCREMENT TO B.

④"A" INPUT IS NOT USED, DATA INPUT IS FED DIRECTLY TO 4 : 1 MUX.

Figure 3-8. Microprogram Counter Block Diagram


FROM CACHE
CAMUX
00 31
I 3 REG
00 15
I 3 REG.
16 - 31
16
I 2 REG.
INSTRUCTION
DECODE
DECODE MACRO INSTRUCTIONS, OP-CODE; AUGMENT CODE, AND SUB-OP-CODE FIELDS INTO μPC ADDRESS
GND
DECODE
SAVE
μPC ADDRESS
μ INTERRUPT
VECTOR FROM
IE-UNIT
08, 09, 10, 11
IE-UNIT DATA STRUCTURE
FOR DATA VECTOR JUMPS
MICRO INTERRUPT RETURN ADDRESS
TRUE
TEST
μPC
CONTROL
A
B
C
D'
E
F
6 : 1
MUX
TEST
TRUE
μ INTERRUPT
CONTROL
CROM 01-03
μPC
PC + 1
OR JUMP
OR BRANCH
ADDRESS TO
CROM
μPC
SAVE
GND
LMICROSEQOE
00 - 15
BUFFER
ISOLATES
μPC DURING
ACS & WCS
READ &
WRITE
μPC
LATCH
LPCLATCHOE
MD BUS
16 31
CROM 48 - 63
16 31
LCRAMADDRCE
FROM
2901
IE-UNIT
YB BUS
00 - 31
WCS
ADDRESS
REG.
16 - 31
LRAMADDR BUS
00 15 TO
DSS CRT
05 - 15
MICROPROGRAM
CONTROL
STORE (CROM)
ADDRESS PROM
4K-64
DATA
PROM-RAM
BANK ENABLE
DECODE
0 4
LEXT
DATA
CE
LRAM
DATA
CE
TO
2901
IE-UNIT
LRAMDATA BUS 00 - 63

6:1 MUX STEER CONTROL

2:1 MUX MICROINTERRUPT CONTROL		4:1 MUX MICRO PC CONTROL					MUX STEER		LHOLD			HSEQ DEFINITION 5 6 0 0 DIN 0 1 ADDER 1 0 STACK 1 1 HOLD								COMMENTS
LSEMICROINT	MICROINTDL	CROM 03	CROM 02	CROM 01	PC Sel B	PC Sel A		SELECTION				4		3		2		1		
									3	2	1	5	6	5	6	5	6	5	6	
0	0	X	X	X	N/A	N/A	DECODE SAVE	B												
0	1	X	X	X	N/A	N/A	MICROINT.	D												CODED 0D08 0DFC
1	X	0	0	0	1	1	CROM	F	0	0	0	1	1	1	1	1	1	0	0	HOP, SEQUENCE 0
1	X	0	0	1	1	1	CROM	F	0	0	1	1	1	1	1	0	0	0	0	LEAP, SEQUENCE 1
1	X	0	1	0	1	1	CROM	F	0	1	1	1	1	0	0	0	0	0	0	BRANCH, SEQUENCE 2
1	X	0	1	1	1	1	YB BUS	E	1	1	1	0	0	0	0	0	0	0	0	L BRANCH, SEQUENCE 3
1	X	0	1	1	0	1	CROM	F	1	1	1	0	0	0	0	0	0	0	0	JUMPDATAORD L BRANCH, SEQ3
1	X	1	0	0	1	1	CROM	F	1	1	1	0	1	0	1	0	1	0	1	NOP, SEQUENCE 4
1	X	1	0	1	1	0	PC SAVE	A	1	1	1	0	0	0	0	0	0	0	0	MICRORETURN, SEQUENCE 5
1	X	1	1	0	1	1	CROM	F	1	1	1	1	0	1	0	1	0	1	0	RETURN, SEQUENCE 6
1	X	1	1	1	0	1	DECODE	C	1	1	1	0	0	0	0	0	0	0	0	JUMP D, SEQUENCE 7

830455

Figure 3-9. Macroinstuction to Microinstruction Block Diagram

enabled and gated to the I2 copy register. The I2 copy register maintains a copy of the right or left half of the I2 register in the IE Unit (the next instruction to be executed). The I2 instruction in the IE Unit is in the decode and effective address calculation phase of execution. The output of the I2 copy register is sent to instruction decode. When I3 contains a fullword or left halfword instruction, I3 (0-15) are gated to I2 (0-15). When I3 contains a right halfword instructions, I3 (16-31) are gated to I2 (0-15). Two cycles are required to empty I3 during halfword instruction sequences.

3.3.2.5.2 Instruction Decode

The instruction decode logic consists of an instruction decode ROM array and an instruction decode multiplexer. The decode logic is driven by the I2 register copy and decodes macroinstruction op-code, augment code, and sub-op-code fields into uPC addresses. The decode array checks the right hand flag, base mode flag, and FPA present flag in determining the uPC address. The relationship between op-codes, modes (flags), and uPC addresses is predefined and programmed (burned) into the PROMs. The output of the PROMs goes to the instruction decode multiplexer which decodes the uPC vector and a copy of the vector is put in DECODESAVE when there is a micro interrupt acknowledged. The decode ROM array can be considered a type of look-up table for op-codes and flags (inputs) versus uPC addresses (outputs). At a minimum, an instruction in the decode phase is two clocks (300 ns) away from execution.

3.3.2.5.3 Address Selection 6:1 Mux

The 6:1 mux, under the control of the uPC control and microinterrupt control, selects the next address to be fed to the microprogram counter. Steer control for the 6:1 mux is listed in table form as part of figure 3-9. The 6:1 mux is constructed of eight dual 4:1 multiplexers and four quad 2:1 multiplexers. Selection of one of the four inputs of the eight dual 4:1 multiplexers is controlled by the three CROM bits (01, 02, 03), the interrupt branch decode latch, and the jump data order; the interrupt branch decode latch must be low to select CROM bits. Selection of micro-interrupt vector or decode save of the quad 2:1 multiplexer is controlled by the output enable LSELMICROINT and the select HMICROINTDL.

Selection A (uPC Save). The uPC save register is used to hold the uPC address of an interrupted microinstruction. The contents of the save register must be gated back through the 6:1 mux to the uPC at the end of the microinterrupt routine. The uPC save register is used as the source of the next firmware vector when the sequence field specifies a six for microreturn.

Selection B (Decode Save). The decode save register saves the uPC address output at the instruction decode. In the event of microinterrupt, the uPC control may select the output of the decode save as the next firmware vector. Sequence field six microreturn follows by JUMPD the next cycle specifies that decode save register is to be used as source to the uPC.

Selection C (Instruction Decode). The instruction decode consists of a decode array and an instruction decode multiplexer. The instruction decode decodes macroinstruction op-code, augment code, and sub-op-code fields into uPC addresses. In addition to the op-code fields the instruction decode also checks the right hand flag, base mode flag, and FPA present flag in determining the uPC address. The output of the instruction decode is gated to the 6:1 multiplexer. Refer to the table on figure 3-9 for mux steering bit combination. An instruction in the instruction decode phase is, at a minimum, two clocks (300 ns) away from execution. Sequence field seven JUMPD indicates that the next instruction is to be decoded to generate the next uPC address to access the RAMs.

Select D (Microinterrupt Vector). The microinterrupt hardware provides a means of interrupting the firmware out of macroinstruction emulation sequences. The microinterrupt must be specifically enabled by the firmware before it can occur. The microinterrupt represents an exceptional event that must be identified and handled by firmware. The exceptional event may be an error, a software interrupt or trap, or may indicate that some hardware element or interlock requires servicing.

The microinterrupt hardware provides a vectorized and prioritized decode of the conditions causing the microinterrupt. The hardware provides a uPC address of a firmware routine designed to identify and handle the exceptional event.

If the exceptional event is an error, a trap, or an interrupt, the firmware will not attempt to return to the sequence in which the microinterrupt occurred, but instead will activate a firmware trap or interrupt handler that in turn activates a software trap or interrupt handler.

If the exceptional event represents a hardware element that needs servicing, the firmware will service that element and resume processing at the point at which the microinterrupt occurred. To implement the microinterrupt, a uPC save register is used to store that interrupted context. MS Unit hardware does not provide for nested microinterrupts.

The microinterrupt input to the 6:1 mux (input D) consists of microinterrupt bits 08, 09, 10 and 11 from the IE unit. When microinterrupt occurs all 4:1 multiplexers will be disabled so that the microinterrupt can be serviced first.

Select E (YB-bus). Input to the 6:1 mux from the YB bus (IE unit data structure for data vector jumps) consists of bits 16 through 31. The uPC control directs the 6:1 mux to select the YB bus input to the uPC during branch (Sequence 3) and jump data order, which usually associate with computed branches or JWCS instruction.

Select F (CROM 48-63). The uPC control directs the 6:1 mux to select the CROM input to the uPC during hop (Seq0), leap (Seq1), branch (Seq2), long branch (Seq3, NOT JUMP data order), and NOP (Seq4) sequences. These types of special branches can activate either 4, 8, 12, or 16 bits of the microword. When the HCROM01 signal is high, the four bit shifter register will provide the select signals LHOLD1 through LHOLD3 that distinguish which of the CROM bits (52 through 63) are used for defining the hop, leap, or branch operations.

3.3.2.6 Micro PC Data Buffer

The uPC data buffer is controlled by LMICROSEQOE and serves as an isolation stage for the uPC during PROM, ACS, and WCS read and write operations.

3.3.2.7 CROM Addressing

Micro PC address bits 00 through 04 are sent to the bank enable decode where the bit combination is decoded and the required bank enable is fed to the PROM, ACS, or WCS bank containing the desired microword. Bits 05 through 15 of the uPC address are used to address the required microword within the selected bank. The microword selected from the microprogram is gated onto the LRAMDATA bus (bits 00 through 63).

3.3.2.8 Control Store

The MS unit control store consist of sixteen 4K x 4 PROMs organized into a 64-bit wide by 4K deep array. The control store is referred to as the control read only memory (CROM). The microprogram located in CROM may be replaced or augmented by a microprogram loaded into WCS (writable control store). Each CROM location (address) contains a microword (see figure 4-1) that provides the basic unit of control for the CPU.

Within the CPU, the microword is used to initiate and monitor the execution of operations. Each microword functions during two microcycles: the CROM cycle and CREG cycle.

- The CROM cycle refers to functions performed at the end of a control store access. These functions are performed by direct output or decodes of the microword without an intervening clock. Typically these functions include microsequencing and monitoring (test) functions.
- The CREG cycle functions refer to functions performed from the control store register (CREG). The CREG cycle functions require an intervening clock and can provide a result oriented operation for a full step (150 ns) following the CROM step. Generally CREG cycle functions control data structure and data put-aways.

  The CPU provides 16 level-orders (8 from the MS unit and 8 from the IE unit). The level-orders may be ordered to a set-or-reset state by firmware (see figure 3-7).

3.3.2.9 Alterable Control Store (ACS)

The alterable control store is comprised of 16, 4K x 4 static RAM elements to provide an array of 4K x 64 bits that allows a shadow control store. ACS may contain a copy of PROM control store, may be written into under software control, and may be the source of the primary execution microcode when the CPU is in an operational state.

ACS provides a base level firmware patch capability that operates under software control. It is, therefore, unnecessary to field replace PROM elements containing factor set firmware.

- ACS Write Instruction. This instruction causes a doubleword to be fetched from memory and loaded into an ACS location. The doubleword from memory is transferred 32-bits (one word) at a time by the RAM data in register to the LMRAMDATA bus that is 64-bits wide.
- ACS Read Instruction. This instruction causes the contents of a specific ACS address (a doubleword) to be gated onto the LRAMDATA bus where it will perform the same duties as a PROM microword.

3.3.2.10 Writable Control Store (WCS)

The WCS consists of two banks of sixteen 4K x 4 static RAM elements to provide an 8K x 64-bit RAM array that is accessed by macrolevel WCS control instructions. Software operates the WCS by the use of the SET CPU (set CPU mode), Read CPU status, write WCS (WWCS), read WCS (RWCS), and jump WCS (JWCS) instructions. (See figure 3-7.)

- The SET CPU (set CPU mode) instruction is used to switch the CPU from operating under the main firmware set (PROM) to ACS. The RD field (bits 6, 7 & 8) of the macroinstruction will specify the GPR in the CPU that will define the operating characteristics of the CPU. If bit 21=1, the CPU will operate under ACS control. If bit 21=0, the CPU will operate under CROM control (default).
- Read CPU status instruction places the current operational status of the CPU into the RD register.
- The RWCS instruction causes the contents at a specific PROM, ACS, or WCS address (a doubleword) to be gated onto the LRAMDATA bus and to the RAM data out registers via the MDB Bus to the MP2901.

  The RWCS memory transfer instruction causes the contents of a PROM, ACS, or WCS location to be stored in memory. The RAM data out register is 64-bits (one word) wide; therefore, two transfers must be made to store a doubleword in memory. The addresses of the WCS location and memory doubleword are provided in two registers (with the uP2901), whose addresses are specified in the RWCS instruction. The RWCS is a privileged instruction.

- The JWCS instruction is an unprivileged fullword instruction using memory reference type format with byte (F-bit) attribute. The byte attribute insures that the final effective address has a linear interpretation of C-bits. The memory reference format allows an index register (field x) or an indirect memory table to be the source of the final effective address.

- The effective WCS addresses provided by the JWCS, WWCS, and RWCS instructions are absolute addresses.

- The MS unit instruction decode may decode an instruction op-code directly into WCS, however the WCS target address and op-code must be predefined and burned into the instruction decode PROMs. Only a subset of WCS addresses are targets of instruction decode operations (see figure 3-2). WCS is used by software via JWCS instruction.

3.3.2.11 Order Structure

The order structure (see figure 3-7) is an output signal array that decodes microwords into individual output signals. From one to four orders may be generated by a single microword. The order structure provides both CROM (control read only memory) and CREG (Control Register) cycle orders and in addition, some orders are delayed for an additional clock to provide CREG +1 cycle orders. In general, orders are used to initialize or terminate internal control functions. Orders may also serve as qualifiers for certain events and register strobe signals. Normally orders are 150 ns pulses; however, the CPU also provides sixteen level-orders that may be ordered to a set-or-reset state by the firmware.

The vertical dimension of the order array (see figure 4-1) is referred to as the order line-number and the microword provides two binary encoded fields (01 and 02) to provide independent order line-number decodes (line addresses). The horizontal dimension of the order array is divided into four order groups. The microword provides four bits (40, 41, 42 and 43) within the OE field to select one or more order groups.

Two orders may be selected simultaneously within a given field (one from each group) if the line numbers of the two required orders are the same. A maximum of four orders (two from each field) may be selected.

The MS unit generates half of the orders in the order structure. Most of these orders are used within the MS unit itself.

Microword bits 48 through 51 are clocked into the control register (see figure 3-7), bits 42 and 43 control the CREG orders decode demultiplexer (bit 42 enables order group 2, and bit 43 enable order group 3). Output CREG orders are listed in figure 3-7 and figure 4-1. Group 2 and group 3 orders are derived from CREG bits 48 through 51.

Microword bits 44 through 47 are clocked directly into the CROM Orders decode demultiplexer, bits 40 and 41 control demux output (bit 40 for order group 0 and bit 41 for order group 1). Output CROM orders are listed in figure FO3-6 and figure FO4-1. Group 0 and group 1 CROM orders are derived from CROM bits 44 through 47. The high state of HCROM 44 insures that orders in the bottm half of order group 0 or order group 1 are to be decoded.

The MS unit also generates eight of the level orders which can be set or reset by firmware. The firmware settings are retained by flip-flops until again processed.

3.3.2.12 Turnkey Panel Interface Logic

The LCPUIPU2SW, LCPU1OFF2SW, LCPU2IPU1SW, and LCPU2OFF1SW inputs from the PROCESSOR SELECT switch on the turnkey panel are routed to a multiplexer/register in the turnkey panel interface. The multiplexer select input is controlled by the HPROCESSOR1L line which is derived from the jumper on the CS Unit. This determines the designations of the two processors as processor 1 and processor 2. The two outputs from the multiplexer provide the LIPU and LOFFLINE signals. These signals are latched and used to qualify which LEDs are driven on the turnkey panel.

The remaining circuitry in the interface includes drivers, buffers, and latches which monitor and drive the CPU console functions and indications. The disable cache (LDSABLCACHE) signal is also latched and routed via the interface.

3.3.2.13 Leading/Trailing Zeros Detect Logic

The purpose of this logic is to examine the Y Bus in order to determine how many leading or trailing 4-bit nibbles contain zeros.

Each one of a series of gates is connected to monitor one 4-bit nibble of the 32-bit word on the Y Bus. The outputs from the eight gates, plus a leading or trailing zeros firmware term (LLDNCTRLEAD or LLDNCTRTRAIL) are used to address a PROM. The PROM produces a 8-bit output corresponding to the number of leading or trailing zeros detected. A 2:1 multiplexer is controlled by a load counter (LLDNCTR) signal, and used to select either Y Bus lines HYB00 through HYB07 or the output from the PROM. The output from the multiplexer is loaded into the N counter (up/down counter with load feature). For a normal load N counter, the multiplexer selects HYB00 through HYB07. For a leading or trailing zeros detect, the multiplexer selects the PROM outputs. The ouput from the counter is routed onto the MDB Bus (HMDB00 through HMDB7) via a buffer controlled by the LNIPLOE signal.

During a nibble shift operation, as the shift takes place, the N counter is automatically decremented.

3.3.2.14 Decode Exceptions (Causes and Vectors)

A decode exception may be considered as any of the interrupt exception cases. Specific causes of an interrupt exception are power fail, system panel attention, CPU halt, interrupt request, IPU trap, and console attention. These conditions generate an interrupt exception (LINTEXCPN) to the microinterrupt vector logic on the IE Unit. Within this logic, the signal is ORed with the IOTRACE+1 signal to produce an interrupt (LINTEXCPNIT). This has a different priority than the normal microinterrupt.

With a valid instruction in I2, signal HI2DECODEVALID is routed from the IE Unit pipeline logic to produce an LI2VALIDINST signal via the MS Unit I2 exception vector logic. If a violated instruction reaches the I2 stage of the pipeline, the HI2REFILLPIPE signal is active. This signal is routed from the IE Unit pipeline logic to the MS Unit and used to qualify the I2 exception vector logic. This is also the case for a map miss (HI2MAPMISS) or an I2 instruction fetch error (HI2INSTERR). These three terms are applied to a priority encoder which generates a group of prioritized encoded error signals (HI2DERR08, 09 and 10) and a decode exception signal (LDCODEEEXCPN). This latter signal automatically inhibits the I2 valid instruction (I2VALIDINST) signal, signifying that the instruction in I2 is invalid. The I2VALIDINST signal going false causes a decode exception to be sensed by controlling a multiplexer. The multiplexer selects the encoded error signals (HI2DERR08, 09, and 10) and the error vector is taken (refer to the following list of decode exception vectors).

VECTOR	DEFINITION
D20	Decode Exception
D40	Pipeline Violation
D60	Decode Instruction Miss
D80	Decode Instruction Error
DA0	Not used
DC0	Not Uused
DE0	Invalid Instruction

3.4 Instruction/Execution (IE) Unit

The IE Unit is the instruction/execution unit. The instruction set-up portion of the board performs effective address calculations and operand prefetching while the execution logic performs the instruction execution steps. The data paths within the IE Unit are 32-bits wide while the control store paths are 64-bits wide. The IE Unit is under microword control which controls, initiates, and monitors the operation of the unit.

The following text is presented in two major subdivisions: IE Unit Components and IE Unit Operation.

3.4.1 IE Unit Components (see figure 3-10)

Note

Items in parentheses (xx) denote Logic Drawing 130-103584 page numbers. These are included as supplemental information for Gould S.E.L. field service personnel. Where applicable, these numbers are included on block diagrams.

3.4.1.1 CAMUX Bus

The cache multiplexer bus (CAMUX bus) provides a 32-bit data path from the cache/SelBUS to the instruction pipeline. Data transmitted on the bus is in the form of operands or instructions.

3.4.1.2 Instruction Cache Register I3 (2)

The I3 register is a 32-bit register that functions as the top level of the instruction pipeline. This register is loaded with the cache output during instruction cache read operations. The output is then sent to the I2 register in the instruction pipeline. A copy of the I3 register is held in the MS Unit.

3.4.1.3 Right Halfword Register (2)

The I3 right halfword register is a 16-bit register that receives the right halfword (bit positions 16-31) from the CAMUX bus during cache instruction read operations. When enabled, bit positions 16 through 31 are gated to I2 (00-15) to perform left shifts of right halfword instructions.

3.4.1.4 I3 Right Halfword and RHW No-Op Detect Logic (2)

This logic detects and indicates that the I3 left or right haflword contains a halfword instruction. It is not valid unless the I3 VALID flag is set. The logic also detects if the right halfword of I3 contains a No-Op code; the resulting signal is used to skip right halfword No-Ops.

3.4.1.5 I2 Register (3)

The I2 register is a 32-bit register that contains the instruction loaded from the I3 register during bump pipe sequences. This instruction is in the decode and effective address calculation phase of execution and is then sent to the I1 register in the instruction pipeline during advance (bump pipeline) sequences.

3.4.1.6 I2 Indirect Register (4)

The indirect register is a 32-bit register that is loaded from the CAMUX bus with the cache output during indirect read operations. The output of the indirect register is gated to the I2 bus in the instruction pipeline.

3.4.1.7 Predecoder (3)

The predecoder comprises three PROMs. It decodes the instruction in I2 to determine instruction attributes.

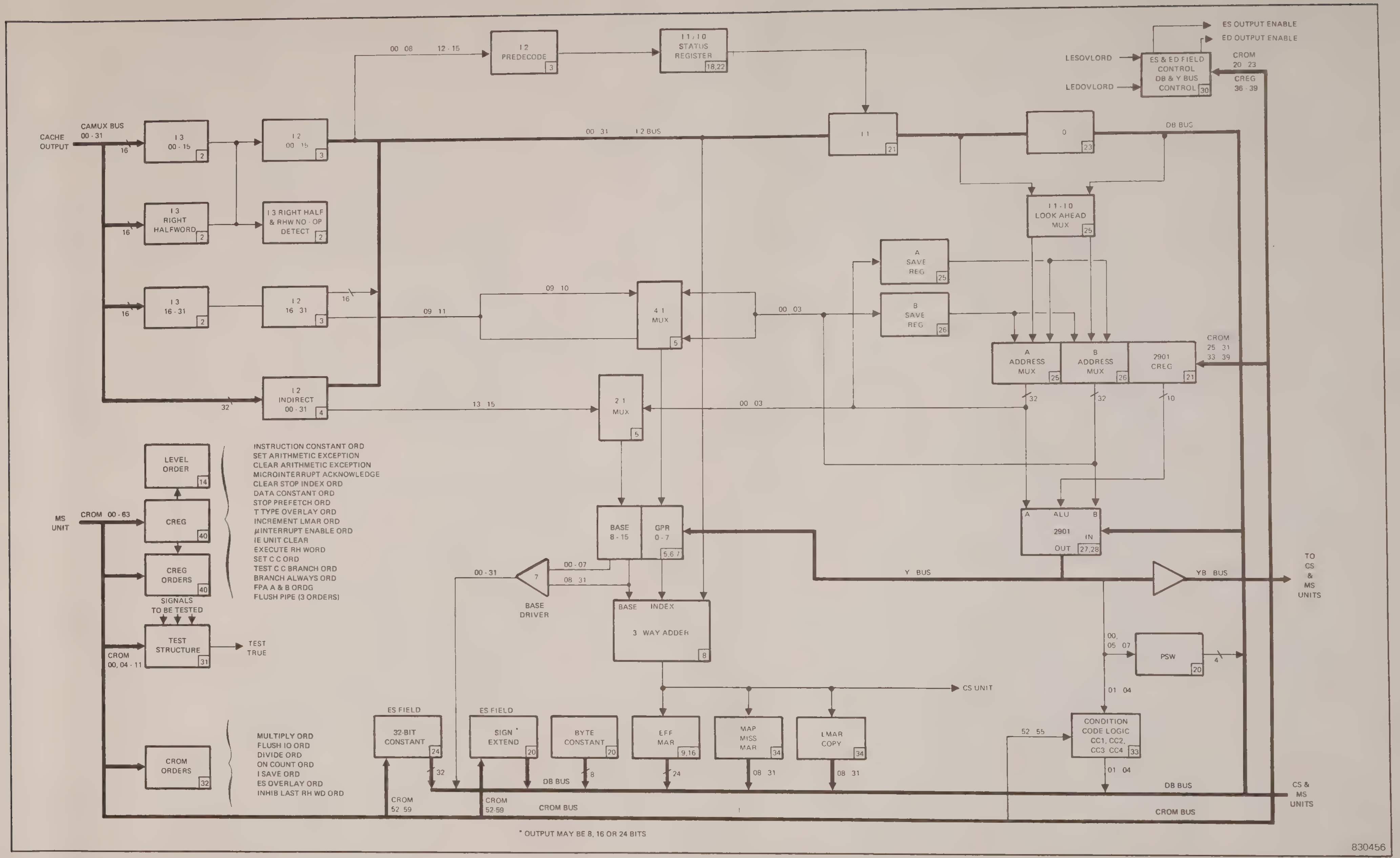

CACHE OUTPUT
CAMUX BUS 00 - 31
16
I 3 00 - 15 2
I 2 00 15 3
00 08
12 - 15
I 2 PREDECODE 3
I 1 / I 0 STATUS REGISTER 18,22
I 3 RIGHT HALFWORD 2
I 3 RIGHT HALF & RHW NO - OP DETECT 2
I 3 16 - 31 2
I 2 16 31 3
I 2 INDIRECT 00 - 31 4
32
00 31
I 2 BUS
09 10
09 11
4 1 MUX 5
13 15
2 1 MUX 5
00 03
I 1 21
0 23
DB BUS
LESOVLORD
LEDOVLORD
ES & ED FIELD CONTROL DB & Y BUS CONTROL 30
ES OUTPUT ENABLE
ED OUTPUT ENABLE
CROM 20 23
CREG 36 - 39
I 1 - I 0 LOOK AHEAD MUX 25
A SAVE REG 25
B SAVE REG 26
A ADDRESS MUX 25
B ADDRESS MUX 26
2901 CREG 21
CROM 25 31 33 39
10
A ALU B 2901 IN OUT 27,28
Y BUS
YB BUS
TO CS & MS UNITS
00, 05 07
PSW 20
4
01 04
CONDITION CODE LOGIC CC1, CC2, CC3 CC4 33
52 55
CS & MS UNITS
CROM BUS
MS UNIT
CROM 00 - 63
LEVEL ORDER 14
CREG 40
CREG ORDERS 40
SIGNALS TO BE TESTED
TEST STRUCTURE 31
TEST TRUE
CROM 00, 04 - 11
CROM ORDERS 32
INSTRUCTION CONSTANT ORD
SET ARITHMETIC EXCEPTION
CLEAR ARITHMETIC EXCEPTION
MICROINTERRUPT ACKNOWLEDGE
CLEAR STOP INDEX ORD
DATA CONSTANT ORD
STOP PREFETCH ORD
T TYPE OVERLAY ORD
INCREMENT LMAR ORD
μINTERRUPT ENABLE ORD
IE UNIT CLEAR
EXECUTE RH WORD
SET C C ORD
TEST C C BRANCH ORD
BRANCH ALWAYS ORD
FPA A & B ORDG
FLUSH PIPE (3 ORDERS)
MULTIPLY ORD
FLUSH IO ORD
DIVIDE ORD
ON COUNT ORD
I SAVE ORD
ES OVERLAY ORD
INHIB LAST RH WD ORD
BASE 8 - 15
GPR 0 - 7 5,6,7
00 - 07
08 31
00 - 31
7
BASE DRIVER
BASE INDEX 3 WAY ADDER 8
CS UNIT
ES FIELD
32-BIT CONSTANT 24
SIGN * EXTEND 20
BYTE CONSTANT 20
EFF MAR 9,16
MAP MISS MAR 34
LMAR COPY 34
8
24
08 31
DB BUS
CROM 52 59
CROM 52-59
CROM BUS
* OUTPUT MAY BE 8, 16 OR 24 BITS
830456

Figure 3-10. IE Unit Block Diagram

3.4.1.8 I1/I0 Status Register (18)

The I1/I0 status register is fed from the I2 predecoder. It is used to generate an instruction valid or not-valid signal. A valid signal allows the pipeline to advance normally. A not-valid signal causes a vector to a dedicated location; no operation is performed but the pipeline is advanced.

3.4.1.9 I2 Bus

The 32-bit I2 bus is used as a unidirectional communication path for instruction decode (vector decode array logic) or processing within the CPU. In addition, it serves as a direct link to the instruction pipeline. I2 register or I2 indirect register.

3.4.1.10 I1 Register (21)

The I1 register is a 32-bit register used to hold the next instruction to be executed. The I1 register is loaded from the I2 register during bump pipe sequences. In dynamic one clock macroinstructions, the contents of the I1 register relates to the micro-PC address held in the micro-PC register during the CROM cycle.

3.4.1.11 Backdate Program Counter (14)

The backdate program counter (PC) consists of a PROM and a sequencer. It is responsible for monitoring and counting the number of instructions loaded into the I3 pipeline register on the IE Unit.

In the case of a halfword instruction or an uncommitted branch, the macro program counter (CS Unit) continues to count. However, the backdate PC is conditioned to recognize that it is not a fullword and does not count. The macro PC may be as many as four instructions ahead of the relevant one in the IE Unit. To locate the required instruction address,the value in the backdate PC is subtracted from the value in the macro PC. In order to accomplish this operation, the read backdate count must always precede the read program status doubleword by one cycle unless the pipeline is frozen.

3.4.1.12 Multiplexer (5)

The multiplexer comprises one 2:1 mux and two 4:1 muxs to select a 4-bit address from the following sources: either the indirect register, the I2 register, or the MP2901 file address registers (E Unit part of IE Unit). The 4-bit address outputs from each of the multiplexers is then routed to access the dual-ported RAM for the corresponding base register and general purpose register. I Unit addresses are examined during the first half of the 150-nanosecond cycle; E Unit addresses are examined during the last half.

3.4.1.13 General Purpose Register (27,28)

Eight 32-bit general purpose registers (GPRs) in the 2901 microprocessor are available for arithmetic, logical, and shift operations. Each register is used as fixed or floating point accumulators, or temporary data storage devices and contain control information such as data address, count, or pointers. Registers 0-7 are the GPRs.

3.4.1.14 Base Register and Index Register (5,6,7)

A dual ported RAM provides sixteen 32-bit registers. Registers 1 through 7 are copies of the GPRs in the MP2901. Registers 8 through 15 provide the base registers for the base register addressing mode of the CPU.

Seven (1-7) of the eight 32-bit base registers are used during effective address calculations (24 bits are used, bits 00 thru 07 are truncated). This is accomplished by adding the contents of the base register and index register to the relative address (machine instruction) to yield the actual memory address (effective and index register address). Registers 8-15 are the base registers.

Seven of the eight 32-bit index registers are used during effective address calculations (24 bits are used). The index registers are I Unit copies of the E Unit GPRs. Registers 1 thru 7 are the index registers in base mode. Registers 1 thru 3 are available as index registers in nonbase mode. Index register 0 is loaded with a zero value during a power-up or system reset sequence. If no indexing is required, index register 0 is used for address calculation.

3.4.1.15 Base Driver (7)

The base driver receives the 32-bit output from the base registers and gates it to the DB bus. It is also used for isolation purposes on the DB bus.

3.4.1.16 3-way Adder (8)

The 3-way adder is used to add the three required arguments for the operand effective address calculation. The arguments consist of the 16-bit or 19-bit displacement (zero extended to 24-bits) from the I-bus, the 24-bit content of the general purpose register, and the 24-bit content of the base register. The output of the 3-way adder is the operand logical effective address which is gated into the logical memory address register (LMAR), on the CS unit, and the effective memory address register (EFFMAR). The EFFMAR output is then sent to the 2901 microprocessor via the DB bus. Any argument not required for the effective address calculation is zeroed at the arguments source. The effective address calculation requires one microcycle (150 nsec).

3.4.1.17 Effective Memory Address Register (9,16)

The three 8-bit effective memory address registers (EFFMAR) are used as temporary storage devices for the 24-bit operand logical effective address from the 3-way adder. The output from the EFFMAR is then gated to the DB bus under control of the E Unit part of the IE Unit.

3.4.1.18 Map Miss MAR (34)

The map miss MAR is loaded by the output of the three-way adder. If an instruction or data fetch results in a map miss, the Map Miss MAR is accessed to provide information used to calculate the required physical address. The relevant map entries are then fetched from cache or main memory.

3.4.1.19 LMAR Copy (4, 34)

The logical memory address register copy (LMAR copy) is a 24-bit register that receives the operand logical effective address from the 3-way adder and gates it to the DB bus. it is used for LA, LEA, and LAB instructions to prevent MAR conflicts.

3.4.1.20 32-bit Constant (24)

The 32-bit constant consists of four 512 by 8 PROMS. Each PROM outputs directly to the DB bus to provide predefined 32-bit constants to the 2901. The address source for the PROM array is either the microword or the I0 register. The I0 path to the constant PROMs is used to generate single bit masks for either the bit in memory or the bit in register macroinstructions.

3.4.1.21 Sign Extend (20)

The sign extend register extends the sign bit (bit 24) of the byte constant. It also sign extends the contents of the I0 register when it contains a halfword. The output of the extension register provides the DB bus with a 32-bit word.

3.4.1.22 Byte Constant (20)

The byte constant logic generates a byte literal for the least significant byte (bits 24-31) of the DB bus. When the byte literal is gated to the DB bus, it is sign extended to provide the DB bus with a 32-bit number.

3.4.1.23 I0 Register (23)

The I0 register provides the bottom level of the instruction pipeline. In dynamic one clock macroinstructions, the contents of the I0 register relates to the microinstruction CREG cycle. The contents of the I0 register is gated to the DB Bus as either a 32-bit fullword or a 16-bit halfword. This register is loaded each time the macroinstruction pipeline is advanced by the firmware.

3.4.1.24 I1/I0 Look Ahead Multiplexer (25,26)

The I1/I0 look ahead multiplexer enables the file addresses to be picked up during the CROM cycle.

3.4.1.25 DB Bus Source Decode

The DB bus supplies the D-input to the 2901 ALU data source selector. The source decode logic decodes a microword field and determines which source register is gated to the 2901 input bus. In turn, this causes a register in either the IE, CS, or MS Unit or the CS unit to be gated to the DB bus. Source decodes are accomplished in the CROM cycle but executed in the CREG cycle.

3.4.1.26 Microprocessor 2901 (27,28)

Eight 2901 microprocessors are used in the 32/67 CPU to provide a full 32-bit parallel path required during SelBUS transfers. Each 2901 is a 4-bit slice containing a 16 location dual ported RAM, an 8 function arithmetic logic unit (ALU), and ALU operand source input multiplexer, and a 4-bit register. In addition, the 2901 provides a bit shift capability, either left or right, and single or double precision formats.

3.4.1.27 Y Bus

The Y bus is a 32-bit data path that is used as the only output from the 2901. The Y bus sends the output to the GPR/index registers (base, index dual-port RAM), on the IE unit, and after being buffered, provides this data to the CS unit and the MS unit.

3.4.1.28 Y Bus Control (30)

This tristate controller determines the destination of data from the MP2901 via the Y Bus using a decode of the microword ED field. Normally, only one logic element can receive data at any time.

3.4.1.29 Program Status Word Register (20)

The program status word (PSW) register is an 4-bit register that contains the privileged bit (bit 0), extended addressing bit (bit 5), enable base register mode bit (bit 6) and the arithmetic exception bit (bit 7). The PSW register is loaded from the Y bus and outputs to the DB bus.

3.4.1.30 Condition Code Logic (33)

The condition code logic consists of two registers, multiplexers, and a 16x48x8 field programmable logic array (FPLA). The condition codes (CCs) are used as software indicators to show the results of an arithmetic and/or logical operation after an instruction has been performed. The condition codes may be sourced from either the Y Bus or the FPLA. The FPLA enables current CCs for the next instruction to be examined (look ahead function).

3.4.1.31 CROM Bus

The 64-bit CROM bus is used to carry the microword from the control store, on the MS unit, to the test structure, order structure, and control register (CREG), on the IE Unit.

3.4.1.32 Test Structure (31)

The test structure is used to monitor external input signals to the CPU and the internal conditions of the CPU. It is capable of performing single or multiple signal tests and provides an indication whether the test is true or false. In the single test operation, each tested signal resides in a different group (horizontal dimension of the array) but contain the same line number (vertical dimension of the array) within each group. In a multiple test operation, the structure identifies either the presence of a specified test (test true) or the absence of all specified tests (test false). In addition, the test structure provides the ability to branch or jump on condition true or condition false.

3.4.1.33 Order Structure (32,14)

The order structure is an output signal array that decodes microwords into individual output signals. They are used to initialize or terminate internal control functions, serve as qualifiers for certain events, and register strobe signals. In addition, the order structure provides both CROM and CREG cycle orders and, if delayed, provides CREG+1 cycle orders.

3.4.1.34 Level Orders (34)

Level orders differ from other firmware orders in that, once set, they remain in effect until reset. The hexadecimal value of CROM bits 52 through 55 determine the level order to be set or reset.

3.4.1.35 Control Register (CREG) (25,26)

The control register provides the necessary registers and gates to transform the 64 control read only memory (CROM) bits, from the MS Unit, into CREG bits for properly sequencing microinstructions. The output from the CREG is to the 2901 microprocessor.

3.4.1.36 Clock Generator (34)

The clock generator is provided for waveshaping and distributing the clock signal throughout the system. The clock signal is generated from the master clock on the SelBUS terminator card at a 150 nsec pulse rate.

3.4.2 IE Unit Operation

The IE Unit performs the instruction/execution steps of the emulation of macroinstructions by the CPU. Primitive macroinstructions such as load, and subtract normally are performed in one clock cycle (150 nsec). The more complex macroinstructions may require many microinstruction steps.

The following description is based on the primary data flow which follows each of the input/output busses within the IE Unit. The text should be read and used in conjunction with figure 3-10, the IE Unit block diagram.

3.4.2.1 CROM Bus Functions

The CROM bus is a 64-bit data bus that routes the microword onto the IE Unit for distribution to the order structure, test structure, the 2901 control register (CREG), and the condition code logic.

3.4.2.1.1 Order Structure

The order structure is an output signal array that decodes microword bits into individual control signals. It provides both CROM and CREG cycle orders.

The last eight line numbers (orders) in order groups 0 and 1 (figure 4-1) are CROM cycle orders. These orders do not require clocking and are coded and generated during each CROM cycle. CROM bits 40 and 41, from the order enable field, selects whether order group 0 or order group 1 will be enabled. CROM bits 44 through 47 select the line numbers for order groups 0 and 1.

The remaining orders are executed either during the CREG cycle, which requires a clock, or the CREG+1 cycle, which requires a delayed clock. CREG cycle orders utilize bits 40 through 43 from the order enable group and bits 44 through 51 from order group 0/1 and order group 2/3.

The order enable field is a group enable for all four groups of orders. CREG bit 40 enables order group 0, bit 41 enables order group 1, bits 42 enables order group 2, and bit 43 enables order group 3.

The order group fields provide the necessary control for all functions outside the data structure that concerns the CPU. CREG bits 44 through 47 select the line numbers for order groups 0 and 1 and CREG bits 48 through 51 select the line numbers for order groups 2 and 3. By including an order enable for all four groups, it is possible to generate four individual orders simultaneously per instruction.

3.4.2.1.2 Level Orders

The level order select field, CROM bits 52 through 55, uses only the P field of the microword P, C, and H fields. The field is enabled, and the level order set, by the LATCH.ORDER. Level orders are reset by simultaneous LATCH.ORDER and LATCH.DATA orders.

3.4.2.1.3 Test Structure (see figures 3-11 and 3-12)

The test structure is capable of monitoring all of the internal conditions and states within the CPU and all of the external signals input to the CPU that affects the units operation. It is capable of performing either a single signal test or a multiple signal test. CROM bits 04 through 11, which comprise the test field, determines which signals are to be tested during the CROM cycle. The test field consists of four individual test group enable bits, bits 04 through 07, and a four bit line number designator field, bits 08 through 11. The horizontal dimension of the test structure matrix is called a group (bits 04 through 07) and the vertical dimension of the matrix is called a line number (bits 08 through 11).

In the single test operation, each tested signal resides in a different group but contains the same line number within each group. In a multiple test operation, the test structure identifies either the presence of a specified test (test true) or the absence of all specified tests (test false). The output from the test structure is routed to the micro-PC on the MS Unit.

3.4.2.1.4 Control Register

The control register (CREG) receives CROM bits 40 through 51 from the order groups and CROM bits 52 through 55 from the P-field of the microword. The 2901 CREG is responsible for transforming the CROM bits from the MS Unit into CREG bits for properly sequencing microinstructions. The output from the CREG is to the 2901 microprocessor.

CROM bits 52 through 55, from the P-field, are used to generate ALU shift codes, condition code select, and latch order select signals.

3.4.2.1.5 Condition Code Logic

Condition codes are used as software indicators to show the results of an arithmetic and/or logical operation after an instruction has been performed. The condition code logic consists of two registers, multiplexers, and a 16 x 48 x 8 field programmable logic array (FPLA).

Condition codes are set in the CREG+1 cycle (the cycle following the CREG cycle). During the execution of instructions, either CREG bits 52 through 55 of the microword or the output of the overlay PROM is selected to determine the condition code rules. The condition code selection is dependent upon the output of the ALU and on the input to the FPLA. The condition code rules are then applicable for that condition code setting. The condition code selection and condition code rules are shown in table 3-1. The branch condition true (BCT), branch condition false (BCF), and branch function true (BFT) instructions allow branching on the condition codes and also testing of the condition codes.

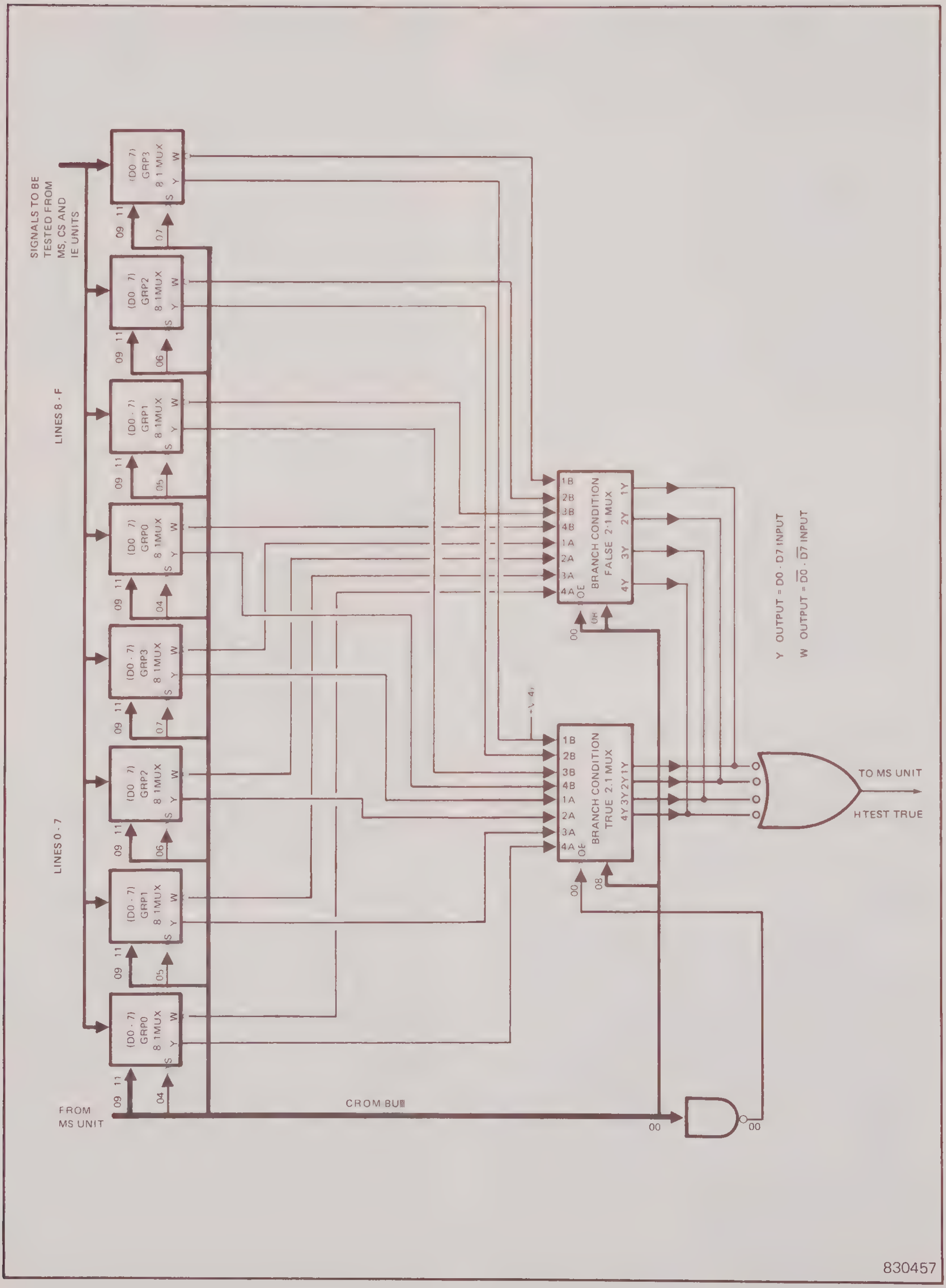

SIGNALS TO BE TESTED FROM MS, CS AND IE UNITS
LINES 8 - F
LINES 0 - 7
(D0 - 7) GRP3 8:1 MUX
(D0 - 7) GRP2 8:1 MUX
(D0 - 7) GRP1 8:1 MUX
(D0 - 7) GRP0 8:1 MUX
S
Y
W
09
11
07
06
05
04
BRANCH CONDITION FALSE 2:1 MUX
BRANCH CONDITION TRUE 2:1 MUX
1B
2B
3B
4B
1A
2A
3A
4A
OE
1Y
2Y
3Y
4Y
00
08
+V 4
Y OUTPUT = D0 · D7 INPUT
W OUTPUT = $\overline{D0}$ · $\overline{D7}$ INPUT
TO MS UNIT
H TEST TRUE
FROM MS UNIT
CROM BUS
00
830457

Figure 3-11. IE Unit Test Structure

TESTS
BITS 4,5,6, and 7

LINE NUMBER

Bits 8,9,10,11	0111 GROUP 0	1011 GROUP 1	1101 GROUP 2	1110 GROUP 3
0000	0 TRUE	0 ALUZ	0 ALUPOS	0 ALUNEG
0001	1 LCROM0	1 ERR. FLAG	1 PRIV	1 MAP
0010	2 SCPATTN	2 D NOTA	2 RETRY	2 BUSY
0011	3 IADDR STOP	3 DADDR STOP	3 N. FPA BUSY	3 AEXP
0100	4 HALT	4 D NOTA1	4 D. TIMOUT	4 D. PE
0101	5 PWR FAIL	5 N. NOVR	5 W. NOTA	5 N.CACHE BUSY L
0110	6 N FPA	6 N. AEPEND	6 MULTI. CYCLE	6 CC TEST
0111	7 TRACE	7 INTREQ	7 INTRENA	7 U PATTN
1000	8 AEXP EN	8	8 EN. LOCK	8 ROMSIM
1001	9 FLAG	9 ADD. SUB	9 EFMAR30	9 N.INT. EXCPN
1010	A READY	A IPU TRAP	A 1030	A RD.EQ0
1011	B I SAVE0	B N.FPA. AEDATA	B 10. REFILL. PIPE	B
1100	C UNBLOCK	C BASE	C EXT	C
1101	D DMAP. MISS	D MAP.INV	D MPC07	D
1110	E N.IPL	E MAP PROT	E N.INDR. MISS	E
1111	F N.IPU	F	F CACHE. HIT	F

0	1 S 3	4 T 11	12 RA 15	16 RB
SEQUENCE		TEST	REGISTER A	REGIST B
0 BCT	0 HOP	BIT 4 = 0, SELECT TEST GROUP 0	2901 BR. FILE FP FILE A-ADDR (0-F)	2901 BR. FIL FP. FIL B ADDR (
1 BCF	1 LEAP	BIT 5 = 0, SELECT TEST GROUP 1	IF MA = 1	IF MA =
2 BRANCH		BIT 6 - 0, SELECT TEST GROUP 2	0 R(R) (10-6,7,8)	0 R(R) (10-6,7,
3 LBRANCH		BIT 7 = 0, SELECT TEST GROUP 3	1 R(RO) (10-6,7,=1)	1 R(R0) (10-6,7,=
4 NOP		BITS 8 - 11, SEE TESTS	2 R(S) (10 9,10,11)	2 R(S) (10-9,10,
5 URTN			3	3
6 RETURN			4	4
7 JUMPD			5	5
			6 R(SO) (10-9,10,=1)	8 BR(BR) (10-6,7,8
			7 BR(IR) (10-13,14,15)	9 BR(BR (10-6,7,=
			8 BR(BR) (10-6,7,8)	A BR(IBS (10-9,10,1
			9 BR(BRO) (10-6,7,=1)	
			F BR(IBS) (10-9,10,11)	
			F BR(BB) (10-13,14,15)	

830458

Figure 3-12. Microword Test Structure

Table 3-1
Condition Code Select

Creg+1 Bits 52 53 54 55	Condition Code Select	Condition Codes	Condition Code Rules
0 0 0 0	SET CC(A)	CC1	Is set if an airthmetic exception occurs
		CC2	Is set if the result (of an ALU function) is greater than zero
		CC3	Is set if the result is less than zero
		CC4	Is set if the result is equal to zero
0 0 0 1	SET CC(AM)	CC1	Retains same code as old CC1
		CC2	Is set if the result is greater than zero
		CC3	Is set if the result is less than zero
		CC4	Is set if the result is equal to zero
0 0 1 0	SET CC(L)	CC1	Always zero
		CC2	Is set if R0-31 is greater than zero
		CC3	Is set if R0-31 is less than zero
		CC4	Is set if R0-31 is equal to zero
0 0 1 1	SET CC(D)	CC1	Is set if an arithmetic exception occurs
		CC2	Is set if (R,R+1) is greater than zero
		CC3	Is set if (R,R+1) is less zero
		CC4	Is set if (R,R+1) is equal to zero
0 1 0 0	SET CC(BIT)	CC1	Is set if R, set bit location (SBL), is equal to one
		CC2	Is set if CC1 was one
		CC3	Is set if CC2 was one
		CC4	Is set if CC3 was one
0 1 0 1	SET CC(Y)	CC1	Is set if Y Bus bit 1 is 1
		CC2	Is set if Y Bus bit 2 is 1
		CC3	Is set if Y Bus bit 3 is 1
		CC4	Is set if Y Bus bit 4 is 1
0 1 1 0	SET CC(M)	CC1	Always zero
		CC2	Always zero
		CC3	Always zero
		CC4	Is set if the result is zero
0 1 1 1	SET CC(LSHIFT)	CC1	Is set if an arithmetic exeception occurs
		CC2	Always zero
		CC3	Always zero
		CC4	Always zero

Table 3-1 (Cont.)
Condition Code Select

Creg+1 Bits 52 53 54 55	Condition Code Select	Condition Codes	Condition Code Rules
1 0 0 0	SET CC (CA)	CC1	Always zero
		CC2	Is set if register D is greater than register S
		CC3	Is set if register D is less than register S
		CC4	Is set if register D is equal to register S
1 0 0 1	SET CC (INDIR)	CC1	Is set if the indirect bit of the word is equal to one and the effective word location (EWL1) in memory is equal to one
		CC2	Is set if (I) is equal to one and EWL2 is equal to one
		CC3	Is set if (I) is equal to one and EWL3 is equal to one
		CC4	Is set if (I) is equal to one and EWL4 is equal to one

During the next ALU function, and dependent upon CREG bits 52 through 55, the new condition codes are selected, set, and interpreted during the next clock cycle.

3.4.2.2 Cache Multiplexer Bus Functions

The cache multiplexer bus (CAMUX bus) routes instructions to the I3 stage of the pipeline, indirect words to the I2 indirect register, and operands to the MS Unit.

The CAMUX Bus is the only direct data input to the instruction pipeline which consists of four serially interconnected registers. The top level of the instruction pipeline I3 and I3 RHW is loaded with the cache output during instruction cache read operations. The I3 right halfword register provides halfword instruction detection and metering. When enabled, bit positions 00 through 15 are disabled and bit positions 16 through 31 are gated to bit positions 00 through 15 in the I2 register to perform left shifts of right halfword instructions.

During the next bump pipe sequence (advance pipeline) the I2 register is loaded. This instruction is in the decode and effective address calculation phase of execution. The decode portion of I2 (I2 predecode) provides a look ahead function to determine whether the current macro-instruction in I2 is a branch, memory reference, memory read or write type instruction.

The I2 indirect register is loaded with the 32-bit cache output during indirect read operations.

3.4.2.3 MAR Polling Logic

The MAR polling logic is responsible for loading the logical memory address register (LMAR) when a memory reference instruction is decoded in the pipeline. As a valid instruction is clocked from I3 to I2, the I2 decode attribute PROMs examine the register to determine whether a memory reference instruction is present. If it is, a decode memory reference (HDCODEMEMREF) signal is generated which provides the enable to maintain the set condition of the poll MAR flip-flop. This flip-flop is normally set, unless reset by an external event, and it generates a load MAR enable (LLDLMAREN) signal. This causes CS Unit logic to clock the contents from the 3-way adder on the IE Unit to the LMAR register, and then onto the LMAR Bus (CS Unit). The LMAR holds the effective logical address while it is converted to a physical address which is used to address the cache memory. The preceding sequence of events is repeated for every memory reference instruction that is clocked into the pipeline.

3.4.2.4 Operand/Transaction Request Logic

During normal operation, a transaction request and a transaction code are routed to the CS Unit (cache) at the start of the first cycle. The transaction is then accepted and completed on the CS Unit, and the IE Unit transaction logic is reset.

The transaction code is derived from the latched outputs of the decode attribute PROM. When the request setup takes place, the PROM outputs are routed to a priority encoder via a transparent latch. They are also routed via an edge-triggered D-type register in parallel with the latch. The priority encoder produces a three-line output which is applied as the B input to a 2:1 multiplexer. The output of the multiplexer is routed to the CS Unit. Firmware generated transaction codes (CREG56 through CREG59) may also be selected via the A input to the multiplexer and routed to the CS Unit on the same lines.

The load logical memory address register enable (LLDLMAREN) signal enables the transaction request logic. A flip-flop, corresponding to the type of transaction request, is set and an operand request (HOPRREQ) signal is generated. This signal is used at the CS Unit to signify that whatever is going to be driven on the transaction lines is valid. HOPREQ also enables the priority encoder. The transaction code is then sent to the CS Unit where it is accepted and completed.

If the transaction is an operand request, an operand output enable (LOPROE) signal is generated, which outputs the contents of the LMAR onto the logical address bus. This address is translated into a physical address by the map and then used to address the cache. LOPROE is also used to reset the transaction logic by disabling the hold paths.

In the event of a microinterrupt, map miss, etc., the transaction codes are saved in the edge-triggered register (in parallel with the transparent latch) by inhibiting the clock. When the microinterrupt has been serviced, the stored transaction code is again presented to the CS Unit. In the case of an operand request, the contents of the LMAR is retained during the microinterrupt and the CS Unit re-executes the transaction.

3.4.2.5 Cache Transfer Type Logic

When an instruction is loaded into the I2 pipeline register, the decode attribute logic (PROM) outputs control signals corresponding to the type of transaction. In the case of cache type transactions, these signals are latched into the cache transfer type logic and encoded to indicate the type of transaction. The codes are routed to cache control logic in the CS Unit. Table 3-2 lists the priorities allocated, the transfer types, and the functions involved in each transaction.

3.4.2.6 Microinterrupt Logic

Microinterrupts are special control signals that serve two functions: shows that an external event has happened that requires attention or that an internal error in the CPU has occurred. The logic necessary to sense microinterrupts is enabled by the microinterrupt signal from the automicrointerrupt logic. Similarly, the order structure outputs three enable order signals which perform the same function. However, these three signals sense certain groups or levels of microinterrupts. For discriptive purposes, these groups are classified as groups A, B, and C.

Table 3-2
IE Unit Cache Transcations

Code/ Priority	Type	Functions
0	Indirect	a. Uses LMAR. b. F and C bits = 0. c. Cache output to I2 indirect register.
1	Odd Read	a. Uses LMAR. b. F and C bits = doubleword. c. LMAR29 = 1. d. Cache output to cache data out register LSW.
2	Instruction Read	a. Used by branch instructions. b. Uses LMAR. c. F and C bits ignored and forced to zero. d. Cache output to I3 register.
3	Store Check Map	a. Uses LMAR and effective F and C bits. b. Causes cache logic to check map for map miss, invalid map, or potential write protect errors. c. Does not write to cache.
4	Read Check Store	a. Uses LMAR. b. Uses effective F and C bits. However, F and C bits are normally forced to zero by a forceword decode attribute c. Cache output to cache data out register MSW. d. Causes cache logic to check map for map miss invalid map, or potential write protect errors.

CONTINUED

Table 3-2
IE Unit Cache Transactions (Cont.)

Code/ Priority	Type	Functions
5	Map	a. Causes a map read operation. b. Uses LMR and effective F and C bits. c. Can produce map errors (miss, invalid, etc.). d. Does not access cache. e. Causes effective physical address to be loaded into the physical MAR. f. Normally coded with a decode attribute that locks LMAR and preserves physical MAR until released by firmware. g. Used by LEAR op code.
6	Data Read	a. Uses LMAR and effective F and C bits (F and C cannot be doubleword, refer to Odd Read). b. Cache output to cache data out register MSW. c. Also used to fetch MSW of doubleword pair. In this case, LMAR29 is forced to zero for even word read.
7	No Operation	a. Causes no cache or map operation. b. Indicates the quiescent state of the set-up logic.

NOTES

Operand request must be true for the cache control logic to accept transaction type (code).

Operand request must stay true until after the transaction code is accepted by the cache control logic.

Table 3-3
Microinterrupt Priority Levels

Priority	Location	Group	Definition
0	D08	A	1) FPA arithmetic exception pending 2) IE Unit arithmetic exception if arithmetic exception enabled
1	D18	A	1) IO TRACE+1 (instruction step) 2) Power fail 3) System panel attention 4) CPU halt 5) Interrupt 6) IPU trap 7) Attention
2	D28	B	1) Data (operand) MAP MISS
3	D38	B	Data operand errors 1) Data time out 2) Data parity error 3) Data MAP invalid 4) Data protect error 5) Data no transfer acknowledge
4	D48	C	1) FPA arithmetic exception pending
5	D58	A	1) Address specification error 2) Data (operand) address stop
6	D68		Not used
7	D78		Not used

Group A, initialized by the LMICROINTAENORD signal, is an overall global event which occurs only once per instruction. This group handles all error including the remaining groups (B and C are subsets of A). Group B, initialized by the LMICROINTBENORD signal, attempts to find an error that is directly related to the instruction being executed. Once the interrupt has been serviced, the CPU returns to that instruction and resumes operations from the stop point. Group B errors are generally data errors or, more specifically, operand related data errors for instructions that perform multiple memory operations. Group C, initialized by the LMICROINTCENORD signal, relates to Floating Point Exception errors and is only used with WCS programs. Table 3-3 shows the priorities, vector locations, group allocations, and definitions of the interrupts.

A priority encoder is responsible for deciphering all the possible errors and then generating the correct microinterrupt vector based on the priority of the errors. The error signals are then ORed together to generate the actual microinterrupt. The microinterrupt signal is routed to the microinterrupt delay latch on the MS Unit. This latch is in the hold state and remains in hold until the next leading 75-nanosecond clock edge. The purpose of the latch is to meet the hold cycle times required by the microsequencer. When the CREG cycle clocks from the CREG clock enable signal, the microinterrupt latched signal and the clock PC generate the H and L select microinterrupt signals. The H select microinterrupt signals causes two 4:1 multiplexers to be disabled. The L select microinterrupt causes a 2:1 multiplexer to be enabled.

When the L select microinterrupt signal is generated, the microinterrupted signal is steered to the B side of the 2:1 mux which generates the actual microinterrupt vector. As this vector is generated, the PC clock, CREG+1 clock, and the CC clocks remain enabled whereas the current CREG cycle clock is disabled. The PC then starts the CROM cycle of the actual microinterrupt. At this point, the CREG clock is disabled for one additional cycle. However, throughout this whole sequence, the execute clocks remain disabled. Then, as the CREG cycle of the actual microinterrupt is performed, the execute clocks are enabled and the execution cycle begins. This flushes the pipeline of all instructions after the one that caused the interrupt.

As the microinterrupt is performed, the control logic on the MS Unit saves the current PC by inhibiting the PC save clock. If the instruction was being clocked into I1 at this time, it is inhibited by the clock decode save but retains the vector associated with the instruction in I1. This is the point at which the microinterrupt active condition is entered. This is the only time an instruction, with this type of microinterrupt, is held in I1 for longer than one cycle. Everything relating to that instruction though, is saved (the address is held in LMAR and the decode vector is held in decode save). The microinterrupt active signal is routed to the IE Unit inhibiting all operand prefetching (effectively freezing the CACHE) and instruction execution until a microinterrupt return signal is returned or a flush pipe is performed.

An operand MAP MISS is the only interrupt from which a microreturn is performed. This allows the CPU to continue operations from the point before the microinterrupt (after the MAP is loaded with the correct data). Any other microinterrupt will flush and refill the pipeline or vector to a TRAP location. If an operand MAP MISS occurs, the CPU holds instruction execution while the operand is fetched and loaded into the MAP. It then reinitializes the transaction logic on the IE Unit. The remaining microinterrupt circuitry is held pending until the transaction is complete. If the operand miss was caused by an indirect fetch, the CPU is unresettable and the pipeline must be flushed and refilled to continue.

On the return, the CROM cycle microinterrupt return signal sets a flip-flop, in the microsequencer logic, as it goes into the CREG cycle. This steers the PC input mux to select the PC save register during the next CROM cycle address. This is the instruction that was interrupted out of. However, if the instruction contains a JUMP D decode, then the decode save register is selected. This utilizes the circuitry previously described except that the signal is steered to the A side of the 2:1 mux (using the H select microinterrupt signal). This causes the next micro PC address to come from the decode save register so that two JUMP D orders are not performed sequentially.

If the CROM cycle advanced the pipeline with a JUMP D order and then the microinterrupt was performed, the return would be back to that CROM cycle. Therefore, anything coded into the CROM cycle has the tendency to be executed twice. To alleviate this, if there is a JUMP D order in the line that is being returned to, the logic prevents the CPU from entering the CROM cycle and goes directly into the CREG cycle.

In a stopclock condition, each stopclock signal is active. The HSTOPCLK1 signal removes the enable to the CREG clock signal H3CREGCLK via the CREG, I/O and condition code logic. Also, the qualify CREG clock (HQUALCREGCLK) signal goes false, and the I1 and I0 pipeline clocks, together with the condition code clock (HCLKCC), are inhibited via the pipeline clocks and CPU stopclock logic.

If a microinterrupt occurs (providing a branch taken condition does not exist), the HMICROINT signal becomes active and the CREG clocks are enabled. As the microinterrupt occurs, a program counter clock is generated at the MS Unit which causes it to accept the vector. The HMICROINT signal is latched to produce an LMICROINTL signal, which is routed via the logic to activate the HQUALCREGCLK signal. This is used to generate a sequence of clocks which control pick up at the microinterrupt address.

The output of the MS Unit is clocked into the CREG cycle. At this time, the condition code clock (HCLKCC) is still inhibited. One cycle later, HCLKCC is enabled and the circuitry is reset to the conditions prevailing before the microinterrupt occurred.

The microinterrupt enable order (LMICROINTCENORD) and the LMAR check busy (LCHCKLMARBUSYORD) are generated from an order decoder that is qualified by a stopclock. The microinterrupt vectors to produce LSTOPCLK and these orders are inhibited.

3.4.2.7 I2 Bus Functions

The I2 bus is a 32-bit unidirectional communication path that feeds the I2 indirect register and the I1 stage of the pipeline.

The I1 register is loaded during the next bump pipe sequence but is advanced automatically one clock after being loaded. This register holds the instructions while the vector target is being accessed during the CROM cycle and while the cache/map is being accessed during an operand request. The output of the I1 register is to the I0 register.

The I0 register holds the instructions during the execution phase. The contents of the I0 register are transferred to the execution logic (2901 DB bus) under firmware control. The firmware controls the following:

- The sign extend feature is used during the execution of immediate type macro instructions and may be sign extended when transferred to the execution logic. Bits 00 through 15 are gated to the DB bus as either all zeros (zero sign extend) or all ones (ones sign extend) according to the state (one or zero) of bit 16 (halfword sign bit).
- Bits 06 through 08 are implemented in an up-counter IC array.
- Bits 06 through 08 may be used to address the software GPRs in the 2901 (registers 0-7). This implements the instruction format RD field (read or write).
- Bits 06 through 08 may also be used for read or write access of base registers 0 through 7 in the dual-ported RAM. This is an alternative interpretation of the instruction format in the RD field.
- Bits 09 through 11 may be used to access the software GPRs in the 2901. This implements the instruction RS field (read or write).
- Bits 09 through 11 may also be used for read or write access of the base registers 0 through 7 in the alternative interpretation of the instruction format RS field.
- Bits 11 through 15 are implemented in a down counter array. Count=0 (minimum) detection logic is used as a firmware test. The firmware test is used during the execution of shift type macroinstructions.

- Bits 13 through 15 are used to address the base registers 0 through 7 in the set-up logic dual-ported RAM. This is a read only access and implements the instruction format BR field.

The I0 register outputs to the DB bus and to the A and B address multiplexers. The A and B address multiplexers supply an input to the A and B input ports of the 2901. These ports are used to address the RAM in the 2901. During microinterrupts the output from the A and B address multiplexers are routed to the A and B save registers. After the microinterrupt sequence, the save registers reroute the data either back into the A and B address muliplexers or to the set-up logic dual-ported RAM.

The dual-ported RAM provides sixteen 32-bit registers. Registers 1 through 7 are copies of the software general purpose registers (2901) and registers 8 through 15 provide the base registers for the base register addressing mode of the CPU.

In the non-base mode three of the eight GPRs may be used as index registers and in the base register mode seven of the eight GPRs may be used as index registers. In either of the modes GPR0 contains all zeros and is used for indexing when no indexing is specified by the macro instruction located in the I2 register.

Of the eight base registers, seven are used as base registers during effective address calculations. During macro instructions that do not specify basing or during the non-base mode operation, the base register output of the dual ported RAM is forced to logical zeros to inhibit basing.

The A and B port register addresses for the RAM may be generated by the I bus instruction or from the microword. Data written into the RAM is sourced from the Y bus and can be read from the B port of the RAM directly to the DB bus. During effective address calculations both the A and B port 24-bit outputs are gated to the 3-way adder.

The 3-way adder is used to add the three required arguments for the operand effective address calculation. The arguments are either the 16 bit or 19 bit displacement (zero extended to 24 bits) from the I bus, the 24 bit content of the index register, and the 24 bit content of the base register. The output of the three way adder is the operand logical effective address which is gated to the effective memory address register and the logical memory address counter register (on the CS Unit). If any argument is not required for the effective address calculation, it is zeroed at the argument source.

The effective memory address register is used as temporary storage device for the output of the three-way adder during the load effective address and load address instructions.

3.4.2.8 DB Bus Functions

The DB bus is a 32-bit data bus that provides the only direct data input to the 2901. Inputs to the DB bus are routed from the effective memory address register, which provides the operand logical effective address, the I0 register, which provides the cache output from the CS unit, or the RAM data out from the MS unit. In addition, the program status word register, the byte constant , and the 32 bit constant registers output to the DB bus. Each of these inputs to the DB bus were previously described.

3.4.2.9 Y Bus Functions

The Y bus is a 32-bit data bus that is used as the only output from the 2901. The data is distributed to the base/index registers (as previously described) and after being buffered, is sent to both the CS Unit and MS Unit.

3.4.3 Microprocessor Operation

The 2901 microprocessor is contained on the IE Unit. Eight of these 4-bit slice microprocessors are used to provide the full 32-bit parallel path required by SelBUS transfers. Each 2901 contains a 16-word by 4-bit dual-ported RAM, a high speed eight function arithmetic unit, and associated circuitry for shifting, decoding, and multiplexing.

Figure 3-13 is a block diagram of the 2901. Each of the major functional elements comprising the microprocessor are discussed in the following paragraphs.

3.4.3.1 Microinstruction Decode

Nine bits of the microinstruction word (figure 3-13) contain three groups of three bits each for selecting the arithmetic logic unit (ALU) internal source operands, the ALU function to be performed, and the resultant shift and internal destinations.

CROM bits 25 through 27, from the internal source (IS) field, select the contents of the registers internal to the 2901. These bits identify the register (R) and sequence (S) combinations, from the R and S fields, that are the source operands.

CROM bits 29 through 31, from the ALU field, determine the eight different ALU functions to be performed.

NOTE

CROM bit 28 comprises a bit position in the ALU function decode but is not considered an integral part of the microinstruction decode. Bit 28 is the carry-in (Cn) bit which is the initial carry-in to the 2901. If an ALU function requires, or uses, the ones complement, the carry-in is not used. If a function requires the twos complement, the carry-in is used.

CROM bits 33 through 35, from the internal destination (ID) field, select the destinations that are internal to the 2901. The internal destinations consist of the RAM register and the Q register.

3.4.3.2 Shifting

Two types of shifters are used to allow for RAM shifting and Q-register shifting. Each of the input data fields to the RAM and to the Q-register are driven by a three input multiplexer and each shifter is controlled from the microinstruction decode inputs, CROM bits 33 through 35 and CROM bits 54 and 55.

The RAM shifter has two ports, RAM 0 and RAM 3, each consisting of a buffer-driver with a three state output and an input to the multiplexer. The three input multiplexer allows the ALU data outputs to be entered, shifted left one bit position (multiplied by two), shifted right one bit position (divided by two), or not shifted in either direction. The output from the two ports is dependent upon the microinstruction decode inputs.

Similarly, the Q-shifter has two ports, Q0 and Q3, which work in much the same manner as the RAM shifter. In the no-shift mode, the multiplexer enters the ALU data into the Q-register. In either the shift up or shift down mode, the multiplexer selects the Q-register data appropriately shifted up or down.

Two sets of 4-to-1 multiplexers support the different possible increase of shifting entries as governed by the microword shift code. One multiplexer is for left shift and the other for right shift. This configuration provides a wraparound for the Q-register or for the sign extend function. CROM bits 54 and 55 from the P-field provide the shift code selections.

Shift overflow logic produces the sign bit on the most significant bit. The exclusive-OR indicates that during the 0 to 31 shifts in the multiply operation, the sign was changed by a shift overflow as noticed by the condition code logic. This is indicated by the shift overflow signal because the sign of the word was changed. Inputs to the 4 to 1 multiplexers provide a sign bit for logical right shift for the division routine.

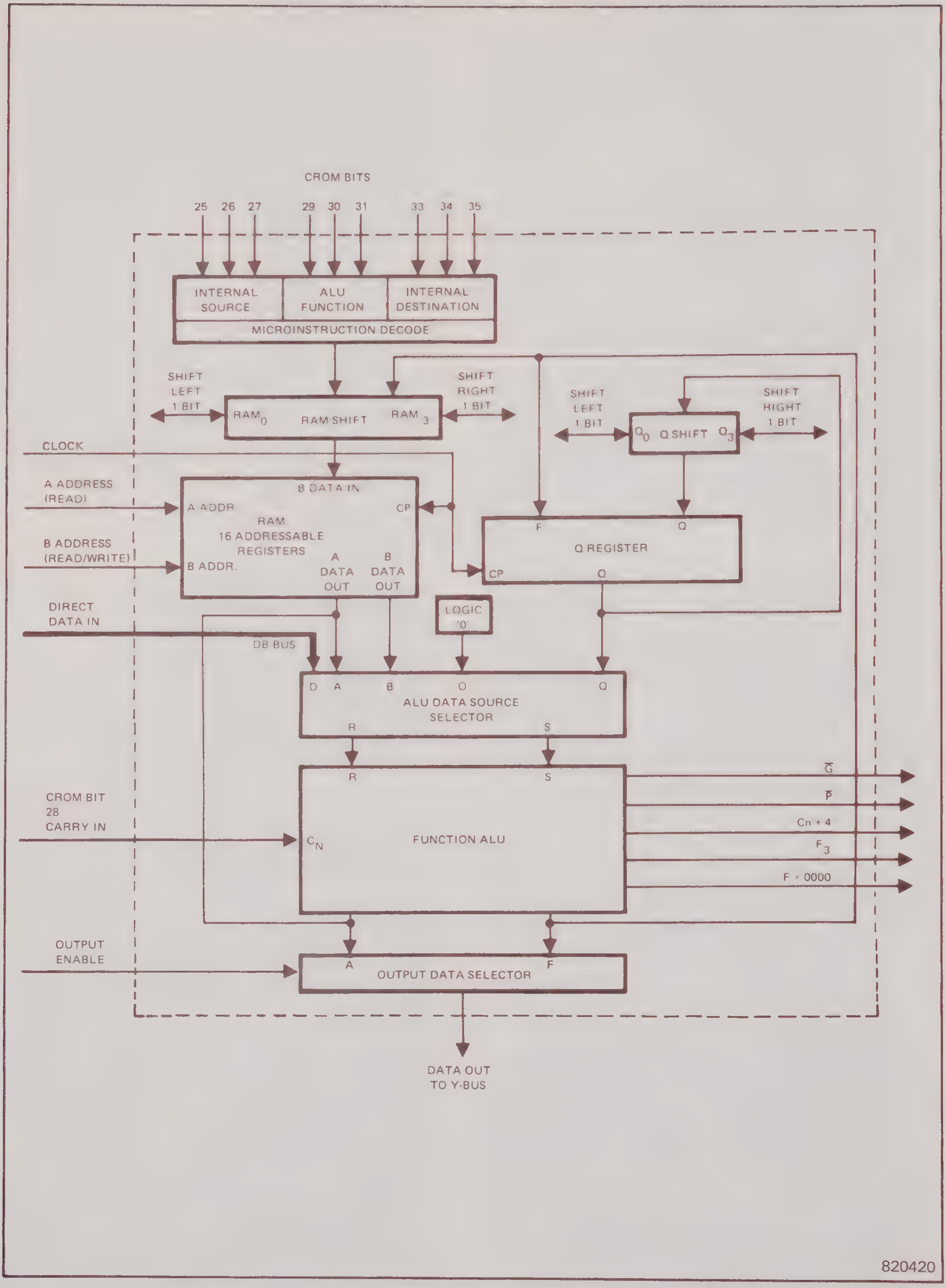

CROM BITS
25 26 27
29 30 31
33 34 35
INTERNAL SOURCE
ALU FUNCTION
INTERNAL DESTINATION
MICROINSTRUCTION DECODE
SHIFT LEFT 1 BIT
RAM$_0$
RAM SHIFT
RAM$_3$
SHIFT RIGHT 1 BIT
SHIFT LEFT 1 BIT
Q$_0$ Q SHIFT Q$_3$
SHIFT RIGHT 1 BIT
CLOCK
A ADDRESS (READ)
B ADDRESS (READ/WRITE)
B DATA IN
A ADDR
CP
RAM 16 ADDRESSABLE REGISTERS
B ADDR.
A DATA OUT
B DATA OUT
F
Q
Q REGISTER
CP
Q
DIRECT DATA IN
DB BUS
LOGIC '0'
D A B O Q
ALU DATA SOURCE SELECTOR
R S
R S
CROM BIT 28 CARRY IN
C$_N$
FUNCTION ALU
$\overline{G}$
$\overline{P}$
Cn + 4
F$_3$
F = 0000
OUTPUT ENABLE
A F
OUTPUT DATA SELECTOR
DATA OUT TO Y-BUS
820420

Figure 3-13. Four-bit Slice Microprocessor Block Diagram

3.4.3.3 Random Access Memory (RAM)

The D-input brings 32 bits of data from the DB bus directly into the ALU data source selector. This input can also be used to modify internal data files. Data in any of the 16 words of the 2901 file can be read from the A-port of the RAM addressed by the 4-bit address field input. Similarly, data in any of the 16 words of the RAM, as defined by the B-address field input, can be read simultaneously from the B-port of the RAM. The same code applied to the A-select field and B-select field causes identical file data to appear at both the RAM A-port and B-port outputs simultaneously. When writing into the RAM, the B-address field defines the destination register of the ALU operation.

3.4.3.4 Q-Register

The Q-register is a 32-bit register used as an accumulator or as a holding register for other applications. However, it is primarily intended for multiplication and division routines.

3.4.3.4.1 Multiplication Routine

The multiplication routine uses the conditional add and shift method. The least-significant bit of the Q-register in the 2901 determines whether the ALU source operands are either A and B (add multiplicand to partial product) or zero and B (add nothing to partial product).

The ALU function in the hardware influences the multiplier to determine whether to do the add/test-and-shift left function or just the shift-left function. When the multiplier bit is one, the A and B is added and then shifted left. When the multiplier and/or multiplicand are zero, for this particular bit, only the B is shifted.

3.4.3.4.2 Division Routine

The division routine conditionally subtracts the divisor from the dividend and then the quotient is shifted right. The equal-to-zero and sign outputs of the 2901 select the appropriate operands for subtraction and shift right. This hardware assist influences the dividend to determine whether a subtract-and-shift right function will occur or just a shift-right function. If subtraction has occurred, a bit is shifted into the least-significant bit of the Q-register to form the partial quotient.

3.4.3.5 Arithmetic Logic Unit (ALU)

The ALU control bit inputs are affected by the CROM word. The CROM bits involved are selected from the entire microinstruction decode order. The outputs from the ALU control registers are the inputs to the 2901 which determines the ALU function to be performed during the CREG cycle.

The ALU is cascadable to a 32-bit wide path over the eight devices by using a look-ahead carry mode. The technique for cascading the ALU sends all carry generate and carry propagate signals of each device to a carry-look-ahead generator.

The ALU is capable of performing three binary arithmetic and five logic operations (refer to table 3-4) on the two 32-bit input words from the data source selector. The R-input field is driven from a two-input multiplexer which provides either the data from the RAM or the direct data inputs from the DB bus. The S-input field is driven from a three-input multiplexer supplied by the A-port or B-port of the RAM or the Q-register. Both multiplexers also have an inhibit capability; that is, no data is passed, which is equivalent to a zero-source operand. The D-input to the ALU is used to allow external registers or data to be input to the 2901.

3.4.3.6 Data Outputs

Data outputs from the 2901 are distributed to several destinations as determined by CROM bits 33 through 35. These outputs are driven externally to the

Table 3-4
ALU Function Control

Mnemonic	Microcode CROM Bits 31 30 29	ALU Function	Symbol (See figure 4-1)
ADD	L L L	R PLUS S	R+S
SUB R	L L H	S MINUS R	S-R
SUB S	L H L	R MINUS S	R-S
OR	L H H	R OR S	R:S
AND	H L L	R AND S	R&S
NOT RS	H L H	NOT R AND S	%R&S
EX OR	H H L	R EX-OR S	R!S
EX NOR	H H H	R EX-NOR S	%(R!S)

Y bus or routed internally for storage in the RAM registers or the Q-register. A two-input multiplexer governs the data output to the Y bus so that either the A-port of the RAM or the ALU is selected.

The RAM data out, which is a 32-bit data word, is routed to the Y bus. The Y bus sends the data to external registers on the IE unit. These GPR's are identical to the registers in the 2901. In addition, the Y bus routes the RAM data out to the CS unit and the MS unit.

The outputs from the ALU consist of the following:

G

P

Cn+4

F3

F=0000

The G and P outputs are the carry generate and carry propagate signals. These signals are sent to the carry-look-ahead generators to speed up the binary arithmetic function.

The Cn+4 output is the carry out signal from the most significant nibble. The output then becomes the carry-in bit, which goes back to the 2901 as the least significant nibble. In addition, it is used as the carry flag in the status register.

The G, P, and Cn+4 outputs are also used for cascading the eight 2901 s.

The F3 output is the sign bit. This signal is the most significant output bit of the ALU and designates the positive or negative result of the ALU operation.

The F=0000 output is the zero detect signal. This signal goes high when all the 2901 F outputs are low, indicating that the resultant ALU operation is zero.

Both the F3 and F=0 outputs are sent to the test structure on the IE Unit.

3.4.3.7 Stopclock Logic

The signals monitored by the stopclock logic are listed in table 3-5 together with the definitions. The logic isuses an HSTOPCLK signal to the MS, IE, and CS Unit to stop appropriate clocks. In some cases, a STOPOPRFETCH signal is also routed to the cache logic in the CS Unit to stop operand prefetching; these are indicated in the table.

Table 3-5
IE Unit Stopclock Conditions

Signal	Definition	HSTOPCLK Generated (Y or N)	STOPOPRFETCH Generated (Y or N)
LCASTOPCLK	Issued when attempt is made to access cache data out, and it is marked invalid.	Y	N
LRDWRSTOPCLK	Isused to stop execution logic during a RAM read or RAM write cycle.	Y	Y
LEMARSTOPCLK	Issued when there is an attempt to load EMAR while previous EMAR has not achieved EMAROE (output enabled).	Y	Y
LCACHEBUSYSTOPCLK	Issued when there is no EMAREOE after a Bus Request E or when there is no OPROE after a Bus Request L.	Y	N
LSTOPFORDATA	Issued to wait for data when a memory reference instruction is valid in the I1 stage but results in a cache miss.	Y	N
LLMARSTOPCLK	Issued to stop the clock for doubleword instructions.	Y	N
LBCKPSTOPCLK	Issued whenever the firmware reads the backdate PC and feeds the count into the MP2901.	Y	N
LMICROINTL	Issued to exit the CPU from the execution step when there are operand related related errors or other external events	Y	Y

Table 3-5
IE Unit Stopclock Conditions (Cont.)

Signal	Definition	HSTOPCLK Generated (Y or N)	STOPOPRFETCH Generated (Y or N)
LBRANCHSTOPCLK	Issued when a branch instruction is in the I0 stage. Stops the CREG cycle of a branch+1 instruction from being executed.	Y	N
LINTORBRANCH+1	Issued one cycle after BRANCHT or when a microinterrupt is detected.	Y	Y
LMRTNSTPFORDATA	Issued when a mciro-interrupt is in the I1 stage and is unable to gain OPROE due to a cache miss.	Y	N
LXTRLSTOPCLK	Issued by the Floating-point accelerator to stop the CPU.	Y	Y

3.5 Cache SelBUS (CS) Unit

The text is subdivided into introductory material, brief descriptions of the major component parts of the CS Unit, and operational information describing the functional interrelationships of the component parts.

3.5.1 Introduction (see figure 3-14)

The major areas of the CS Unit are the map, the cache, and the SelBUS interface. The primary communication busses are the SelBUS, the YB Bus, the DB Bus, the logical memory address bus (LMAR Bus), and the CAMUX Bus. The SelBUS is the external port. The YB bus and the DB bus are the two primary micro-level busses between the CS unit and the IE unit and form the internal port. The YB bus is the output of the 2901 microprocessor. The DB bus is the external input to the 2901. The LMAR bus routes the logical address generated by the set-up functions on the IE Unit (from the 3-way adder). The bus output of the cache to the set-up logic is the cache multiplexer (CAMUX) bus. The CAMUX bus carries operands or instructions.

The CPU operates in two different modes which are, in essence, determined by the firmware. When the CPU is in the idle loop, its operation may be considered in terms of a stand-alone microprocessor environment. When the CPU is actively processing instructions and is no longer in the idle loop, all three units (MS, IE, and CS) operate separately.

Primarily, in the microprocessor environment, the YB bus and the DB bus input and output the cache-in function. This function uses the EMAR (execution logic)

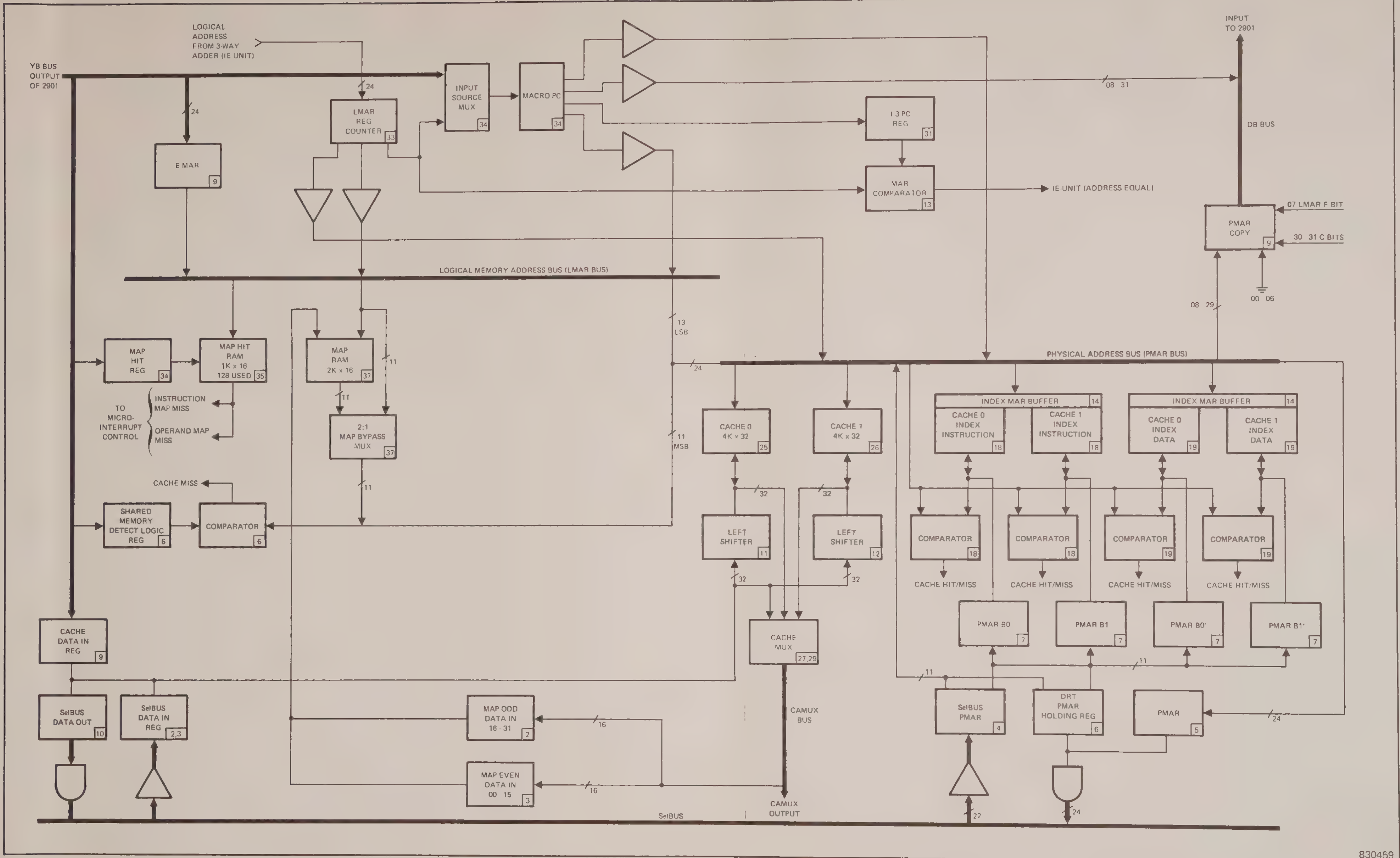

LOGICAL ADDRESS FROM 3-WAY ADDER (IE UNIT)
YB BUS OUTPUT OF 2901
24
24
E MAR 9
LMAR REG COUNTER 33
INPUT SOURCE MUX 34
MACRO PC 34
I 3 PC REG 31
MAR COMPARATOR 13
IE-UNIT (ADDRESS EQUAL)
08 31
INPUT TO 2901
DB BUS
PMAR COPY 9
07 LMAR F BIT
30 31 C BITS
00 06
08 29
LOGICAL MEMORY ADDRESS BUS (LMAR BUS)
PHYSICAL ADDRESS BUS (PMAR BUS)
13 LSB
11 MSB
24
MAP HIT REG 34
MAP HIT RAM 1K x 16 128 USED 35
MAP RAM 2K x 16 37
11
11
2:1 MAP BYPASS MUX 37
11
TO MICRO-INTERRUPT CONTROL
INSTRUCTION MAP MISS
OPERAND MAP MISS
CACHE MISS
SHARED MEMORY DETECT LOGIC REG 6
COMPARATOR 6
CACHE 0 4K x 32 25
CACHE 1 4K x 32 26
32
32
LEFT SHIFTER 11
LEFT SHIFTER 12
32
32
INDEX MAR BUFFER 14
CACHE 0 INDEX INSTRUCTION 18
CACHE 1 INDEX INSTRUCTION 18
INDEX MAR BUFFER 14
CACHE 0 INDEX DATA 19
CACHE 1 INDEX DATA 19
COMPARATOR 18
COMPARATOR 18
COMPARATOR 19
COMPARATOR 19
CACHE HIT/MISS
CACHE HIT/MISS
CACHE HIT/MISS
CACHE HIT/MISS
PMAR B0 7
PMAR B1 7
PMAR B0′ 7
PMAR B1′ 7
11
11
CACHE DATA IN REG 9
CACHE MUX 27,29
CAMUX BUS
CAMUX OUTPUT
SelBUS DATA OUT 10
SelBUS DATA IN REG 2,3
MAP ODD DATA IN 16 - 31 2
MAP EVEN DATA IN 00 15 3
16
16
SelBUS PMAR 4
DRT PMAR HOLDING REG 6
PMAR 5
24
22
24
SelBUS
830459

Figure 3-14. CS Unit Block Diagram

on the CS Unit. If a memory write transaction or an I/O type transaction via the SelBUS is required, the EMAR is used to write an address. Then, for a write transaction, the data is routed to the CS unit on the YB bus. This data is passed through the cache data in register and is loaded into the SelBUS data out register; the address goes through the map if required. Eventually, the data and address are output onto the SelBUS (data on the data bus, address on the destination bus).

Cache transfer type logic is contained in the IE Unit. Table 3-2 (IE Unit description) lists the transfer types, the priorities allocated, and the functions involved in each transaction. The transaction codes are routed to the cache control logic in the CS Unit.

A memory write transaction can be mapped or unmapped as designated by specific transaction codes generated by the microprocessor (i.e., a derivation of the microword). The data is also looked-up in the cache and, if present, it is updated. If the data is not present, it is put into the cache and simultaneously written to the main memory via the SelBUS.

If an I/O-type transaction is requested, mapping and the cache are not envolved. However, access to the cache is inhibited while the I/O transaction goes through.

The macro PC is used as the address source for an instruction read. For a data read, either the LMAR or the EMAR is used as the address source. The address may or may not be mapped. If the operand is found in the cache, it is read from the cache and output onto the DB bus via the cache data out register (MS Unit).

If the cache is empty, or if a cache miss occurs, a cache control line is used to signify that the data or instruction must be read from main memory. A SelBUS transaction is then generated which routes the physical address (mapped or unmapped) to the PMAR and the DRT PMAR. The address is output onto the SelBUS destination bus together with appropriate SelBUS tags which define a memory read operation. After the data read transfer, a data return transfer (DRT) is generated. The physical address is decoded at the cache interface logic and the data on the bus is gated into the SelBUS data in register. This data is routed through the left shifters and the 3:1 cache multiplexer and loaded into the cache data out register. The data is simultaneously put into the cache and transferred to the IE Unit via the cache data out register and the DB bus (MS Unit).

The PMAR is always loaded, regardless of whether the data is output on the SelBUS or not. However, the PMAR copy register is only loaded for operand reads and operand writes. When a transaction is routed via the PMAR to the SelBUS, the address is copied into the DRT PMAR holding register. The corresponding data return from the memory is accompanied by physical address tags and not a whole physical address. Therefore, when data is returned, the address held in the DRT PMAR is used to drive PMAR B0 and PMAR B1 for the cache put-away cycle.

3.5.2 CS Unit Components

The following paragraphs provide brief descriptions of the major component parts of the CS Unit.

NOTE

> Items in parentheses (xx) denote Logic Drawing 130-103516 page numbers. These are included as supplemental information for Gould S.E.L. field service personnel. Where applicable, these numbers are also included on block diagrams.

3.5.2.1 EMAR (9)

The CS Unit effective memory address register is fed from the DB Bus. It is used to address the RAM arrays (cache banks, index RAMS, and valid RAM) via the LMAR Bus and the PMAR Bus.

3.5.2.2 Macroprogram Counter (34)

The macro PC is a 22-bit counter register oriented to a word boundary. It is incremeted by one every time an instruction word is fetched from main memory or cache.

3.5.2.3 Input Source Multiplexer (34)

The address that is loaded into the macro PC may be from either the YB bus or the logical memory address register (LMAR) bus. The 2:1 input source multiplexer is used to determine the source. During context switching (load map time) and load program status doubleword sequences, the address is loaded from the YB bus. The LMAR bus is used as the source for branch instructions.

3.5.2.4 I3 PC Register (31)

The I3 PC register holds the address of the instruction which is currently in the I3 register on the IE Unit.

3.5.2.5 LMAR Counter (33)

The logical memory address register (LMAR) counter receives the 24-bit output of the three-way adder on the IE Unit and holds the effective logical address while it is converted to a physical address and used to address the cache memory. It is used as a counter to access sequential logical addresses as in the load or store doubleword instruction and the load or store file instruction. The LMAR counter is a word oriented register and its apparent width is controlled by the CPU addressing mode. The output of the LMAR counter may be routed to the macro PC via the 2:1 input source multiplexer during branch instructions.

3.5.2.6 MAR Comparator (13)

The memory address register (MAR) comparator compares the contents of the I3 PC register with the contents of the LMAR counter. If the compared logical addresses are the same, and the macroinstruction being executed is a store, the macroinstruction pipeline will be violated. In this case, the output signal from the comparator is routed to the IE Unit and used to refetch the information in the I3 register after the store.

3.5.2.7 Map RAM (37)

The 2K by 16-bit map RAM functions as the primary element for converting logical addresses into physical addresses. A four-megaword logical address space is provided in mapped-extended or mapped-based address mode.

3.5.2.8 Map Hit RAM (35)

The map hit RAM contains 1024 16-bit locations; however only 128 locations are used. Each bit in the map hit RAM corresponds to a register in the map RAM and indicates whether the register has been loaded or not. The hit RAM is reset during each change of map contents.

3.5.2.9 Map Bypass Multiplexer (37)

The map bypass multiplexer is used to select either the 11 most significant bits of the MADDR bus (physical address) or the 11-bit output of the map RAM. The selected 11 bits are appended to the 13 least significant bits of the PADDR bus to form the physical memory address bus (PMAR bus).

3.5.2.10 Cache Memory (25,26)

The total cache memory comprises sixteen 4K by 4 RAM ICs, providing an 8K word cache. The cache is divided into two 4K word banks, cache bank 0 and cache bank 1. Each bank is addressed in parallel by the 11 least significant physical memory address bits from the PMAR bus (plus one instruction/operand bit).

3.5.2.11 Cache Index RAMs (18,19)

Each cache bank has an associated cache index RAM which tracks the storage of memory words in any given location. There are four 4K by 12 cache index arrays, two assigned to instructions and two assigned to data. The index RAMs are addressed in parallel by the 11 least significant physical memory address bits from the PMAR bus. The contents of each index RAM is a valid bit and the most significant 11 bits of a physical word address.

3.5.2.12 Cache Data In Register (9)

The 32-bit cache data in register receives cache write data from the IE Unit via the YB bus.

3.5.2.13 Left Shifters (11,12)

There is one left shifter associated with each of the two cache banks. The 32-bit shifters are required during cache writes to align data to byte, halfword, or word boundaries. The data received by the left shifters is always right justified and comes from either the SelBUS data in register or the cache data in register. The SelBUS data in register is used as the source during DRT copies into cache. The cache data in register is the data source for CPU cache writes.

3.5.2.14 Index MAR Buffers (14)

Two 11-bit memory address (MAR) buffers are provided, one for each bank of index RAMs. Both buffers drive the low-order 11 bits of the physical memory address during writes to the index RAMs.

3.5.2.15 Cache Comparators (18,19)

The four cache comparators compare the high-order 11 bits of a 22-bit physical word address with the output from each of the four index RAMs. This is used to determine in which cache bank a requested memory word is located.

3.5.2.16 Cache Multiplexer (27,28)

The 3:1 cache multiplexer is used to route data onto the CAMUX bus from either of the two cache banks or from the SelBUS data in register.

3.5.2.17 Shared Memory Detect Logic (6)

In a system with two CPUs, the area of memory from 128K up to the maximum of four megawords may be shared. Upper and lower shared memory limits are defined by software. A comparator associated with the shared memory detect logic compares this limit information with the PADDR bus. If an access to shared memory is detected, the output from the comparator is used to provide a cache miss signal to the cache control logic in the CS Unit and the cache is bypassed. The requested information is fetched from SelBUS memory one word at a time.

3.5.2.18 SelBUS Data Out Register (10)

The SelBUS data out register holds data during I/O and memory write transfers. The register is loaded from the YB bus (via the cache data in register). The data is gated to the SelBUS (data bus) under control of the SelBUS interface logic.

3.5.2.19 Physical Memory Address Register (PMAR) (5)

The PMAR holds the physical memory addresses or I/O device addresses during output transfers to the SelBUS. The PMAR is loaded from the PADDR bus and the addresses are gated to the SelBUS (destination bus) under control of the SelBUS interface logic.

3.5.2.20 Data Return Transfer (DRT) PMAR (6)

The DRT PMAR holds a physical memory address while a memory read is in progress. The contents of the register is used to address the cache when the DRT is copied into it. The DRT PMAR is loaded from the PMAR and its contents may be gated to the PADDR bus.

3.5.2.21 SelBUS Data In Register (2,3)

The SelBUS data in register receives data from the SelBUS during DRTs. If the DRT is a response to a memory read, the contents of the register is copied into cache. If the DRT is a response to an I/O SelBUS transfer, the contents of the register is transferred through the cache data structure to the cache data out register (MS Unit) and out to the IE Unit on the DB bus. In this case, the cache is not altered.

3.5.2.22 SelBUS PMAR (4)

The SelBUS PMAR receives the memory address of an externally generated SelBUS memory write, and this is used to address the cache. If the addressed word is in the cache, it is invalidated. The SelBUS PMAR is loaded from the SelBUS (destination bus), and its contents may be gated to the PADDR bus to address the cache.

3.5.2.23 Map Data In Registers (2,3)

There are two 16-bit map data in registers, designated odd and even. They are loaded in parallel with one 32-bit memory word from the SelBUS (data bus). The contents of each register may be individually gated to the map RAM write data lines during firmware map load sequences.

3.5.2.24 PMAR Copy Register (8)

The PMAR copy register is loaded from the PMAR bus whenever an operand address is converted from logical to physical and loaded into the PMAR. The contents of the PMAR copy register is routed to the IE Unit, via the DB bus, during load real address instructions. This address copy is also used during an operand map miss sequence.

3.5.3 CS Unit Operation (see figure 3-14)

The functional interrelationships between the major component parts of the CS Unit are described in the following paragraphs. The description is subdivided by logical function into effective address logic, memory management logic, cache memory logic, and SelBUS interface.

3.5.3.1 Effective Address Logic

The effective address logic comprises the hardware registers, counters, and address necessary to compute and store both operand and effective logical addresses and instruction logical addresses.

The instruction logical address logic comprises the macroprogram counter (macro PC) and an associated 2:1 input source multiplexer. The macro PC is a 22-bit counter register oriented to a word boundary. It is incremented by one each time an instruction word is fetched from main memory or cache. The address that is loaded into the macro PC may be from either the YB bus or the logical memory address (LMAR) bus; the input multiplexer is used to determine the source. During context switching (load map time) and load program status doubleword sequences, the address is loaded from the YB bus. The LMAR bus is used as the source of branch instructions.

A majority of the operand effective address logic is contained in the IE Unit and is described in the text beginning at paragraph 3.4. However, the logical memory address register counter (LMAR reg counter) and the memory address (MAR) comparator are contained in the CS Unit. Also, the cache multiplexer bus (CAMUX bus) originates in the CS Unit.

The LMAR register counter receives the 24-bit output of the three-way adder (on the IE Unit), via the LMAR bus, and holds the effective logical address while it is converted to a physical address and then used to access the cache memory. It is used as a counter to access sequential logical addresses as in the load or store doubleword instruction and the load or store file instruction. Using the register as a counter obviates the need to recompute the effective logical address. The LMAR counter is a word oriented register and its apparent width is controlled by the CPU addressing mode. The width may be either 17 bits for a 19-bit logical address mode or 22 bits for a 24-bit extended logical address mode. The output of the LMAR counter may be routed to the macro PC (via the 2:1 input source multiplexer) during branch instructions. For branch instructions, the LMAR counter first contains the target address then, after incrementing, it contains the target address plus one word. The memory address register (MAR) comparator compares the contents of the I3 PC register with the contents of the LMAR counter register. If the logical addresses are the same, and the macroinstruction being executed is a store, the macroinstruction pipeline will be violated. In this case, the output signal from the comparator is routed to the IE Unit (I3 and I2 status register pipeline violation circuitry) and used to refetch the instruction in the I3 register after the store.

3.5.3.2 Memory Management Logic

The memory management logic provides the hardware necessary to convert logical addresses into physical addresses. The hardware comprises the following functional elements: the map RAM, the map hit RAM, and the map bypass multiplexer.

The 2K by 16-bit map RAM functions as the primary element for converting logical addresses into physical adresses. Each map entry contains a valid bit, four write protect bits, and an 11-bit map block address. The 11-bit map block address facilitates 2K word map blocks; each map block provides 2K word granularity. Each of the four write protect bits provides a 512-word write protection granularity. Since the map RAM is 2K words deep, a four-megaword logical address space is provided in mapped-extended or mapped-based address modes.

The map RAM is addressed via the map address bus (MADDR bus). Data to be written into the map RAM is routed from the cache output or via the SelBUS and two map data in registers (odd and even). Each of these 16-bit registers contains a single map entry fetched from the main memory under control of firmware contained on the IE Unit. Since this is a word fetch, two map entries are brought in each time.

Conversion from a logical address into a physical address is performed by addressing the map RAM with the most significant 11 bits of a 24-bit logical address. The output produced is an 11-bit physical map address which is appended to the 13 least significant bits of the logical address. The resultant 24-bit address is used to access cache or main memory.

Context switch time (load map time) is minimized by organization which uses a look-aside buffer technique. The map hit RAM is the look-aside buffer.

The map hit RAM contains 1024 16-bit locations; however, only 128 locations are used. Each bit in the map hit RAM corresponds to a register in the map RAM and indicates whether the register has been loaded or not. During each change of map contents, the hit RAM is reset. When the map is subsequently accessed, the corresponding hit RAM bit is not set. This causes a map miss microinterrupt to be routed to the IE Unit. The required pair of map entries are fetched from main memory or cache and the corresponding bits in the hit RAM are set.

Bit checking of the hit RAM is accomplished by using the most significant 11 bits of a 24-bit logical address. The first seven bits address one of the 128 locations. The next four bits select one of the 16 hit RAM output bits.

The hit RAM is addressed via the MADDR bus. Data may either be written to the hit RAM over the YB bus or via bit generaton logic (in the first case, via the map hit register). The YB bus is the data source when the hit RAM is reset or when an initial configuration is loaded into it. The bit generation logic array is essentially a single-bit generator (one of 16) which causes one bit to be ORed into the current contents of a hit RAM word. The bit generation array is used to update single locations during dynamic updates of the map RAM locations. The single output of the hit RAM is the miss signal. This is separated into an instruction map miss signal and an operand map miss signal and routed to the IE unit microinterrupt logic or decode exception logic.

The map bypass multiplexer is used to select either the 11 most significant bits of the MADDR bus (logical address) or the 11-bit output of the map RAM. The selected 11 bits are appended to the 13 least significant bits of the PADDR bus to form the physical memory address bus (PMAR bus). The MADDR bus is selected via the multiplexer whenever the CPU is unmapped or temporarily unmapped due to IE unit firmware I/O transfers or physical memory accesses. At all other times, the bypass multiplexer selects the map RAM output to generate the PMAR bus. The PMAR bus is routed to the following locations: the four cache RAMs, the four cache hit comparators, the physical memory address register (PMAR), the PMAR copy register, and the data return transfer (DRT) PMAR holding register.

3.5.3.3 Cache Memory Logic

The cache memory logic provides a high-speed copy of relevant portions of standard SelBUS memory. The cache hardware comprises the following functional elements: cache RAMs, cache index RAMs, index RAM comparators, cache data in registers, left shifters, index MAR In buffer, cache multiplexer, right shifter (MS Unit), cache data out odd and even registers (MS Unit), indirect register (IE Unit), I3 register (IE Unit), data return transfer (DRT) PMAR holding register, and shared memory detect logic.

The total cache memory comprises sixteen 4k by 4 RAM ICs, providing an 8K word cache. The cache is divided into two 4K word banks, cache bank 0 and cache bank 1. Each bank is addressed in parallel by the 11 least significant physical memory address bits from the PMAR bus (plus one instruction/operand bit).

Each cache bank has an associated cache index RAM which tracks the storage of memory words in any given location. There are four 4K by 12 cache index arrays, two assigned to instructions and two assigned to data. Again,the index RAMs are addressed in parallel by the 11 least significant physical memory address bits from the PMAR bus. The contents of each index RAM is a valid bit and the most significant 11 bits of a physical word address.

In order to determine if a specific memory location is cache resident, a 22-bit physical word address is presented to the cache and index RAMs. The lower-order 11 bits of this address select the same location in a group of six RAMs (two cache and four index RAMs). The high-order 11 bits of the address are routed to four comparators where they are compared with the output from each of the index RAMs. If the comparison is good (equal), and if the selected index RAM location is marked valid, the requested memory word is located in the cache within the bank determined by the index RAM that generated the equal comparison. If no equal comparison is found on any index RAM output, the requested memory word is not in the cache.

In order that the cache contains a reliable copy of SelBUS main memory, each externally originated SelBUS memory write must be monitored and used to invalidate the cache if the specified memory byte, halfword, or word is in the cache. For the same reason, CPU memory-writes to cache must also be written to SelBUS memory. For CPU byte or halfword-writes to cache and main memory, if the word containing the byte or halfword is not in cache, the cache will not be changed. However, the byte or halfword will be written to memory.

During normal operation, starting from a reset state, the valid bits in the index RAMs are reset to zero. When the CPU begins to execute instructions, operands and instructions are read from the cache. However, a cache miss occurs causing the control logic to read memory using SelBUS memory read transfers (MRTs). The requested data is returned from memory in SelBUS data return transfers (DRTs). Each DRT received by the CS unit is stored in the cache. The 11 most significant physical memory address bits are stored in the corresponding index RAM location and, if the memory location did not indicate an error (parity error or uncorrectable data error), the corresponding valid bit in the index RAM is set. When the cache is updated, the requested data is simultaneously routed to the instruction pipeline register (I3 register on the IE unit) or to the operand holding register (cache data out registers on the MS Unit).

To increase the speed of the cache fill-up sequence each time a cache miss occurs on a read access, the control logic reads a doubleword pair from SelBUS memory and stores the doubleword in the cache. The doubleword is fetched using back-to-back memory read transfers (MRTs) to the even and odd memory word locations. The control logic always reads doublewords even though the cache access may have been for a byte or for a halfword. Operand alignment is performed (if required) as the data is read out of the cache to the cache data out registers. One exception is if the CPU accesses a memory location in a shared memory block; in this case, the control logic only reads one word.

With a 4M-word main memory and a 4K-word cache, there are 1024 possible memory words that can exist in any given cache location (either operand or instruction). The cache bank can hold only two of the 1024 possible alternatives. When a particular cache location in both banks is valid, and the CPU accesses a third memory location for the given cache location, a conflict exists as to which of the two banks the third location will be loaded into. This conflict is resolved in the least recently used (L.R.U.) circuitry which records cache access to the two banks. Whenever a cache bank is accessed, a flag is set to indicate that access. Another flag, corresponding to the other bank, is reset. When a conflict occurs, the flags are sampled and the cache bank indicating least recent use is loaded. Whenever a choice between banks is made, the states of the flags are changed.

Each of the cache banks (0 and 1) are 4K by 32-bit RAMs. Within each bank, there is a 2K operand cache and a 2K instruction cache. Each location contains one 32-bit memory word. A cache bank is addressed by the 11 least significant bits of the PADDR bus, excluding the two C bits of the PADDR bus. data is written to, or read from, the cache banks on a common input/output bus. Data may be written to the cache either via the SelBUS data in register or via the cache data in register through one of the two banks of left shifters. The SelBUS data in register is used to copy data return transfers (DRTs) into the cache. The cache data in register is used for source data during CPU writes to the cache.

An instruction is read from the cache through the cache multiplexer to the pipeline I3 register (on the IE unit). Data is read from the cache through the cache multiplexer and the right shifter (MS unit) to the cache data out odd/even registers (MS unit).

In a system with two CPUs, the area of memory from 128K up to the maximum of 4 megawords may be shared. The actual area to be shared is defined by software in mimimum blocks of 128K, and it is implemented by information routed in on the YB bus to the shared memory detect logic register. The information determines an upper and a lower memory limit. The associated comparator compares this information with the PADDR bus. If an access to a shared memory block is detected, the output from the comparator is used to provide a cache miss signal to the cache control logic on the CS unit and the cache is bypassed. The requested information is fetched from SelBUS memory one word at a time.

3.5.3.4 SelBUS Interface

The SelBUS interface logic performs the following main functions:

- Monitors the SelBUS for externally initiated memory writes.
- Establishes SelBUS transfer priority.
- Drives memory and I/O transfers out to the SelBUS and monitors the SelBUS response signals.
- Receives transfers from the SelBUS (DRTs) and monitors the error response line.

The primary function is to monitor the SelBUS for externally initiated memory writes. This function ensures that the cache contains a reliable copy of SelBUS memory. The SelBUS memory address of each externally initiated memory write is captured and a cache access at this address is initiated. If a cache hit is identified, the corresponding cache index RAM location is invalidated.

Transfer priority is established in one of three ways depending upon the transfer type (I/O or memory) and the CPU mode (CPU or IPU). I/O transfers are made at the highest SelBUS transfer priority (0). If an attempt by the cache to access the SelBUS is not granted, priority 0 is allocated to it. Memory transfers (read or write) can be made at either pseudo priority 23 or SelBUS priority 22, depending upon the CPU mode and the priority selection jumper located in the SelBUS interface logic. The CPU, or one of the CPU/IPU pair, operates at pseudo priority 23 which signifies the absence of another SelBUS device polling at transfer priorities 0 thru 22. In this mode, the SelBUS interface logic is polling each time a CPU-originated cache access occurs. If a cache miss is indicated, a SelBUS transfer (read or write) will be executed on the next SelBUS clock. The SelBUS memory write transfer occurs whether or not a cache hit was found.

In a CPU and IPU configuration, one of the pair must operate at SelBUS transfer priority 22. The interface operating at priority 22 cannot poll the SelBUS for transfer priority until one clock after the cache access. Therefore, this memory cycle time is one clock longer. A jumper in the SelBUS and polling transfer logic determines the priority regardless of whether the processor is operating as CPU or IPU. The jumper in position selects priority 22, and the associated processor is referred to as processor 2. For memory read transactions, a code is returned with the operand or instruction which signifies whether it is an odd or even word transfer. The codes are as follows:

PROCESSOR 1
80 - operand even word fetch
82 - operand odd word fetch
81 - instruction even word fetch
82 - instruction odd word fetch

PROCESSOR 2
84 - operand even word fetch
86 - operand odd word fetch
85 - instruction even word fetch
87 - instruction odd word fetch

A third function of the SelBUS interface is to format and execute the requested SelBUS transfer (assuming SelBUS transfer priority has been established). If SelBUS transfer priority has not been established (i.e., won), the interface withholds the transfer for an available SelBUS transfer cycle. After a SelBUS transfer cycle has been executed, the interface monitors the SelBUS response lines to determine if the transfer was accepted by the addressed SelBUS device. For I/O transfers, the interface informs the IE unit firmware of the state of the transfer as follows:

accepted, rejected (addressed channel not present), rejected (channel busy), rejected (channel retry).

If a memory write transfer is accepted, the transfer cycle is complete. If a memory read is accepted, the interface conditions itself to receive a data return transfer (DRT). If a memory read or write is rejected due to non-present memory, the error is reported to the IE unit depipeline logic and the IE unit test structure. If the memory is busy, and the transfer is rejected, the interface samples the SelBUS echo lines. These lines are used to select one of four SelBUS memory inhibit lines which indicates a continuing memory busy condition. When the selected memory inhibit line goes false, the Interface automatically retrys the the memory transfer, including re-establishing SelBUS transfer priority.

The fourth function of the SelBUS interface is to receive DRTs from I/O and memory SelBUS devices. For I/O DRTs, the contents of the DRT is transferred through the cache data structure to the data cache register. At this stage, the data is available to the IE unit. The contents of an I/O DRT is not stored in the cache. With memory DRTs, the contents of the DRT is stored in the cache, and the corresponding index RAM is marked valid only if the SelBUS has not indicated that the DRT contained a memory parity error or an uncorrectable data error. If a memory error has occurred, the index RAM location is marked invalid and the memory error signal is routed to the IE unit depipeline logic and test structure.

The SelBUS interface data structure contains the following functional elements: the SelBUS data out register, the physical memory address register (PMAR), the DRT PMAR, the SelBUS data in register, the SelBUS PMAR, the map data in odd and even registers, and the PMAR copy register.

The SelBUS data out register holds data during I/O and memory write transfers. The register is loaded from the YB bus (via the cache data in register) the data is gated to the SelBUS (data bus) under control of the interface logic.

The PMAR holds the physical memory addresses or I/O device addresses during output transfers to the SelBUS. The PMAR is loaded from the PADDR bus and the addresses are gated to the SelBUS (destination bus) under control of the interface logic.

The DRT PMAR holds a physical memory address while a memory read is in progress. The contents of the register is used to address the cache when the DRT is copied into it. The DRT PMAR is loaded from the PMAR and its contents may be gated to the PADDR bus.

The SelBUS data in register receives data from the SelBUS during DRTs. If the DRT is a response to a memory read, the contents of the register is copied into the cache. If the DRT is a response to an I/O SelBUS transfer, the contents of the register is transferred through the cache data structure to the cache data register and out to the IE unit on the DB bus. In this case, the contents of the cache is not altered.

The SelBUS PMAR receives the memory address of an externally generated SelBUS memory write, and this is used to address the cache. If the addressed word is in the cache, it is invalidated. The SelBUS PMAR is loaded from the SelBUS (destination bus), and its contents may be gated to the PADDR bus to address the cache.

There are two 16-bit map data in registers, designated odd and even. They are loaded in parallel with one 32-bit memory word from the SelBUS (data bus). The contents of each register may be individually gated to the map RAM write data lines during firmware map load sequences.

The PMAR copy register is loaded from the PMAR bus whenever an operand address is converted from logical to physical and loaded into the PMAR. The contents of the PMAR copy register is routed to the IE unit, via the DB bus, during load real address instructions. The contents of the PMAR copy register are also used during an operand map miss sequence.

3.6 Instruction Pipeline

The instruction pipeline is a direct channel for information through which machine instructions pass from source to user. The source being defined as a location in memory and the user being the 2901 Microprocessor.

Minimum cycle time for completion of an instruction through the entire pipeline is four clock cycles. During these four clock cycles; a minimum of eight operations for the memory reference instructions need to be accomplished. For descriptive purposes, each of the eight operations are categorized into four different stages of the pipeline. These are: Backing Store Stage (I3), Decode Stage (I2), Vector Stage (I1), and Execute Stage (I0). Figure 3-15 is a simplified block diagram outlining the different stages of the pipeline and in which clock cycle each operation occurs. The I3 and I2 stages may be considered as asynchronous in that instructions are continually fetched into the pipeline. The I1 and I0 stages are synchronized to the execution phase of the instruction. The following lists the operations that need to be performed per each stage:

1. Backing Store (I3)
    a) Load I3 from CACHE memory
2. Decode (I2)
    a) Perform Instruction Decode
    b) Compute Effective Address
3. Vector (I1)
    a) Access the decode address in the PROMs or perform the CROM cycle of the Control Store Array (vector target)
    b) Translate the physical address
    c) Perform the operand fetch
4. Execute (I0)
    a) Execute the instruction
    b) Fetch another instruction

3.6.1 Backing Store Stage (I3)

The backing store stage is the top level of the instruction pipeline. Normally, this stage of the pipeline is used as a holding register for the instructions. However, it also provides halfword instruction detection and right halfword no operation (NOP) detection.

The CAMUX bus clocks the instruction into the I3 registers on both the MS unit and IE unit. Full word instructions are gated to the two 16 bit I3 registers on the IE unit and the two 16 bit I3 copy registers on the MS unit. During the next clock cycle, this instruction is advanced to the decode stage (I2) of the pipeline. The MS copy of this instruction only gates the left halfword to the decode stage. Two halfword instructions can be packed in one fullword. During the backing store sequence, either a fullword instruction or two halfword instructions may be loaded into the I3 register. A predecoder monitors the output of the I3 register to determine if it is a fullword or a halfword pair. In the case of a fullword, all 32 bits are advanced to the I2 register. For a halfword pair, the left halfword (bits 00 thru 15) is advanced to the I2 register first. The next time that the I3 register is enabled, the right halfword (bits 16 thru 31) are loaded into the I2 left halfword register (bit positions 00 thru 15). When the I3 register contains a left halfword and a no operation (NOP) instruction in the right halfword, the sequence is different. In order to avoid a NOP during the next half cycle, if the halfword is valid, the I3 right halfword NOP detect logic decodes the I3 right halfword. If the right half of the I3 halfword contains a NOP opcode and a NOP sub op code, a signal is sent to the I3 halfword detect PROM. This signal is used to skip a right

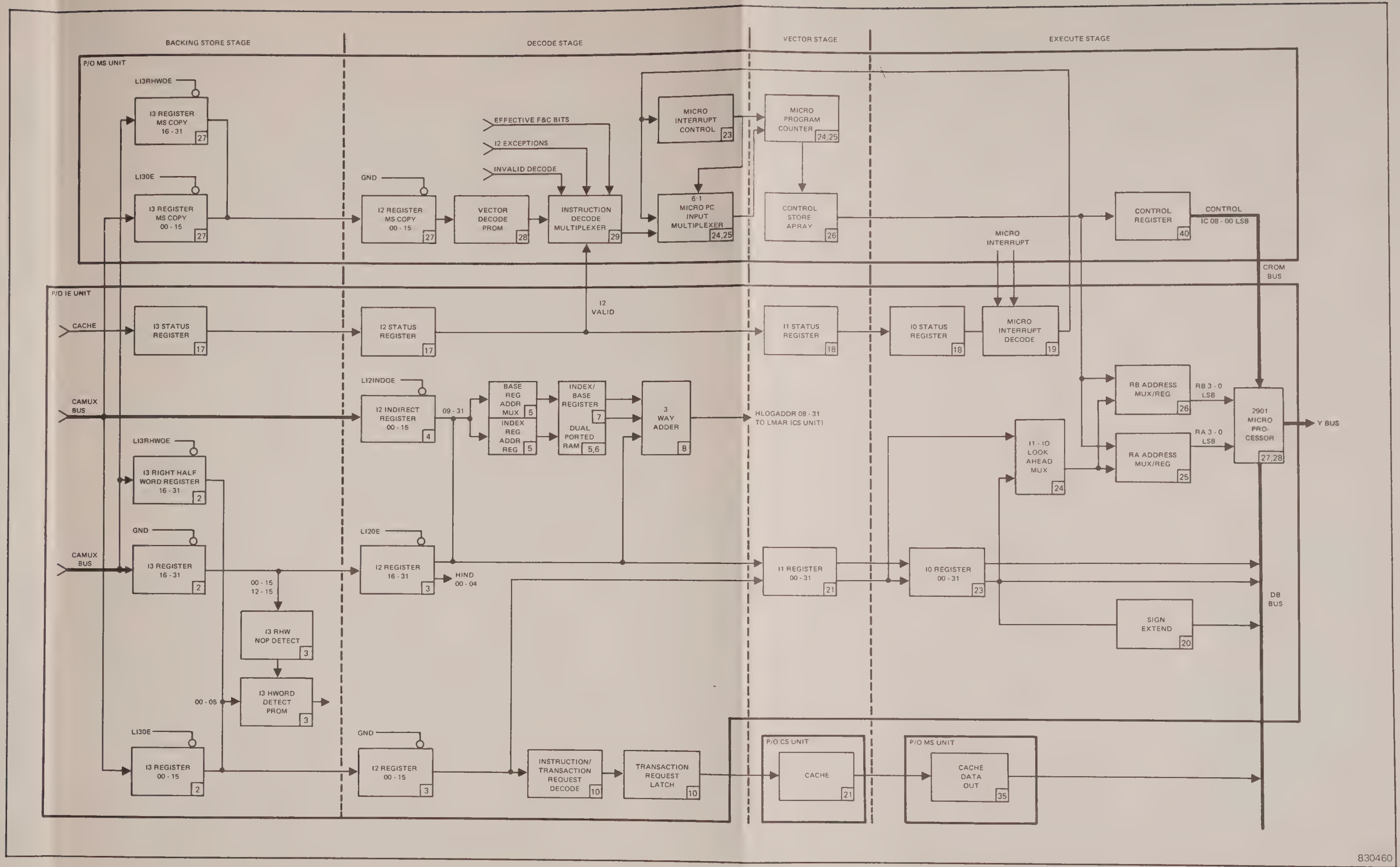

BACKING STORE STAGE
DECODE STAGE
VECTOR STAGE
EXECUTE STAGE
P/O MS UNIT
LI3RHWOE
I3 REGISTER MS COPY 16 - 31 27
LI3OE
I3 REGISTER MS COPY 00 - 15 27
GND
I2 REGISTER MS COPY 00 - 15 27
VECTOR DECODE PROM 28
EFFECTIVE F&C BITS
I2 EXCEPTIONS
INVALID DECODE
INSTRUCTION DECODE MULTIPLEXER 29
MICRO INTERRUPT CONTROL 23
6:1 MICRO PC INPUT MULTIPLEXER 24,25
MICRO PROGRAM COUNTER 24,25
CONTROL STORE ARRAY 26
MICRO INTERRUPT
CONTROL REGISTER 40
CONTROL IC 08 - 00 LSB
CROM BUS
P/O IE UNIT
CACHE
I3 STATUS REGISTER 17
I2 STATUS REGISTER 17
I2 VALID
I1 STATUS REGISTER 18
I0 STATUS REGISTER 18
MICRO INTERRUPT DECODE 19
CAMUX BUS
LI2INDOE
I2 INDIRECT REGISTER 00 - 15 4
09 - 31
BASE REG ADDR MUX 5
INDEX REG ADDR REG 5
INDEX/ BASE REGISTER 7
DUAL PORTED RAM 5,6
3 WAY ADDER 8
HLOGADDR 08 - 31 TO LMAR (CS UNIT)
RB ADDRESS MUX/REG 26
RB 3 - 0 LSB
RA ADDRESS MUX/REG 25
RA 3 - 0 LSB
2901 MICRO PRO- CESSOR 27,28
Y BUS
I1 - I0 LOOK AHEAD MUX 24
LI3RHWOE
I3 RIGHT HALF WORD REGISTER 16 - 31 2
GND
CAMUX BUS
I3 REGISTER 16 - 31 2
LI2OE
I2 REGISTER 16 - 31 3
HIND 00 - 04
I1 REGISTER 00 - 31 21
I0 REGISTER 00 - 31 23
DB BUS
SIGN EXTEND 20
00 - 15 12 - 15
I3 RHW NOP DETECT 3
00 - 05
I3 HWORD DETECT PROM 3
LI3OE
I3 REGISTER 00 - 15 2
GND
I2 REGISTER 00 - 15 3
INSTRUCTION/ TRANSACTION REQUEST DECODE 10
TRANSACTION REQUEST LATCH 10
P/O CS UNIT
CACHE 21
P/O MS UNIT
CACHE DATA OUT 35
830460

Figure 3-15. Instruction Pipeline Block Diagram

halfword NOP. The I3 halfword detect PROM then determines whether or not the halfword is valid. The following conditions must be met in order for the halfword to be advanced through the pipeline:

- I3 valid flag is set in the I3 status register
- HI3HWORD signal is true
- HI3LHWORD signal is false

Table 3-6, I3 Instruction Attribute Decode, lists the signal names and describes their functions that qualify either a fullword instruction or halfword instruction. Once all the conditions are satisfied, and during the next clock cycle, either the fullword or halfword instruction is advanced to the decode stage of the pipeline.

When the I3 register transfers that instruction, the cache is then enabled and routes another instruction into the I3 register. This is accomplished by the macro program counter (macro PC). The macro PC provides a look ahead function that points to the next sequential address, for a cache fetch during the next cycle. The cache then routes the data to the Backing Store Stage of the pipeline. This sequence of events allows the pipeline to remain full at all times which speeds up the execution of instructions. The cache always prefetches an instruction unless it is busy. A signal is returned from the IE Unit, to the cache, to signify that the prefetch has been accepted together with the increment macro PC enable signal. If this enable signal is not returned, the same instruction is refetched.

Table 3-6
I3 Instruction Attribute Decode

Signal	Function
LI3NOINDEX	I3 Halfword Detect PROM decodes for opcode F4 which represents the Branch after incrementing by a Byte, Halfword, Word, and Doubleword instructions. Inhibits indexing and index register conflict checks during non-indirect or first cycle indirect sequences at the decode stage
LI3RHWNOP	Decodes I3 right halfword. Indicates right half of I3 has a NOP opcode and a NOP sub opcode. Signal is used to skip right halfword NOPs.
LININVLDRHW	Decoded only when I3 right halfword register is gated to I3 Bus bits 00-15. Indicates that the opcode present is invalid. Normally, identifies fullword opcodes found in the right halfword.
HI3HWORD	Indicates that I3 left or right halfword contains a halfword instruction. Not valid unless the I3 valid flag is set.
HI3LHW HI3LHWQUAL LI3LHWQUAL	Indicates that I3 contains a left and right halfword pair. The left half is next to be gated to decode stage. Not valid unless I3 valid flag is set.

NOTE

If I3 is valid, HI3HWORD is true, and HI3LHW is false, then I3 contains a right halfword that must be gated to I2

3.6.2 Decode Stage (I2)

The decode stage of the instruction pipeline performs the decode and effective address calculations for the remainder of the pipeline. Both the effective address calculation and the I2 decode steps are performed in parallel to enhance system operation.

The I3 registers from the backing store stage of the pipeline transfer the instruction into the two 16 bit I2 registers during the next clock cycle. Also during this time, bits 00 through 15 are clocked into the I2 copy register on th MS unit.

The I2 registers in the decode stage perform a three fold operation: decodes instructions, requests an operand prefetch, and addresses the set-up logic.

The I2 predecoder determines the type of instruction to be executed and from where the instruction is processed. Branching instructions, load/store instruction, and memory write instructions are processed from the two 16 bit I2 registers. Memory indirect read operations are performed from the I2 indirect register. The output from the registers are gated to the vector stage (I2) for processing.

Simultaneously, the outputs from the I2 registers address the base and index registers of the IE Unit GPR copy (dual ported RAM). During an effective address calculation, the contents of the base and index registers are gated to the three way adder, depending upon whether the instruction is nonbased or based. In the nonbased mode, general purpose registers (GPRs) 1, 2, and 3 output a 24 bit address (bits 8-31) to the 3-way adder. The actual GPR selected is dependent upon bits 9 and 10 from the I2 register. A 19-bit offset value (bits 13-31), from the I2 register, is also routed to the 3-way adder. The 3-way adder sums the two values and outputs either a 24 bit logical address, if extended mode is used, or a 19 bit logical address, if nonextended mode is used.

In the base mode, all seven of the GPRs (1-7) may be used as index registers and are selectable dependent upon bits 9,10, and 11 from the I2 register. A 24 bit address is routed from the GPR's to the 3-way adder. The base register also provides a 32-bit address to the 3 way adder with the first eight bits truncated. However, the offset value is only 16 bits wide and the upper 8 bits are zero extended. These three values are summed in the 3-way adder which outputs a 24-bit logical address. In base mode, all eight base registers (0-7) are available, but only seven (1-7) are available for base address calculation.

The following describes the properties associated with the non base and base modes and extended and nonextended modes:

- Non based - non extended - non mapped mode
    - logical address equals physical address
    - maximum address for instruction or data is 128 KW
    - no write protect
- Non based - non mapped - extended mode
    - logical address equals physical address
    - maximum instruction address is 128 KW
    - maximum data address is 4 MW
- Non based - non extended - map mode
    - maximum logical address for instruction or data is 128 KW
    - maximum physical address is 4 MW
    - MAP Page is 2 KW granularity
    - write protection is 4 blocks per MAP page
    - write protect block is 512 word granularity
- Non based - mapped - extended mode
    - maximum logical instruction address is 128 KW
    - maximum logical data address is 4 MW
    - maximum physical address is 4 MW
    - write protect is 4 blocks per page at 512 W per block

- Non mapped - base mode
    - logical address equals physical address (4 MW)
    - does not support indirect addressing
    - seven GPRs (1-7) are available for memory indexing
    - seven base registers (1-7) are available for base effective addressing calculation
    - no memory write protection
- Mapped - Based
    - logical address equals physical address (4 MW)
    - does not support indirect addressing
    - seven GPRs (1-7) are available for memory addressing
    - seven base registers (1-7) are available for base effective address calculations
    - memory write protect is 4 blocks per page at 512 W per block

The I2 register, bit positions 00-15, also outputs to the instruction/transaction request decode logic. If an operand request is contained within the instruction, the logic decodes it and request an operand fetch. This request is then latched into the transaction request latch and, during the next clock cycle, accesses the CACHE.

The MS copy of the I2 register contains bit positions 00-15. Basically, this register is used as a temporary holding register for the instruction decode logic. The instruction decode logic, which contains the vector decode, decodes the macro-instruction op-code, augment code, and sub-op-code fields into micro program counter (uPC) addresses. The decode hardware decodes the uPC vector. If a valid instruction is being passed through the pipeline, the I2 register sends a valid signal to the decode multiplexer. The decode vector then accesses the uPC input multiplexer. However, a not valid signal causes a vector to a dedicated location. No operation is performed, but the pipeline is advanced. As soon as a valid signal is received from the I2 register, the decoder vectors to the normal location. Table 3-7, I2 Instruction Attribute Decode, lists the signal names and describes the function of each signal.

The micro PC input multiplexer (6:1 mux) receves the address from one of six sources: uPC save register, decode save register, YB-Bus (output from 2901), or from the control store (CROM). The output of the 6:1 mux, under the control of the uPC control and microinterrupt control, selects the next address to be fed to the microprogram counter.

The decode stage outputs are transferred to the vector stage during the next clock cycle.

3.6.3 Vector Stage (I1)

The vector stage of the pipeline is the beginning of the CROM cycle for the execution of instructions. This portion of the logic calculates the physical address, performs the operand fetch, accesses the decode information in the PROMs, an performs the CROM cycle of the Control Store Array (vector target).

During the vector stage, the logical address is output to the MAP RAM on the CS unit. Conversion from a logical to a physical address is performed by addressing the MAP RAM with the most significant 11 bits of the 24-bit logical address. This output is then appended to the 13 least significant bits to form the 24-bit physical memory address. This address then accesses the CACHE or Main Memory.

An operand fetch is performed in the same manner as the instruction fetch. However, once the MAP RAM is accessed, the output is asserted to the cache multiplexer on the CS unit. Both the vector stage and execute stage of the pipeline are in the wait mode until the operand is fetched. When the data is returned, it is written into the cache and simultaneously loaded into the cache data out register on the MS unit (via the cache multiplexer and the CAMUX bus).

The Micro Program Counter (uPC) on the MS unit is responsible for selecting decoded information from the PROMs and performing the CROM cycle of the control

Table 3-7
I2 Instruction Attribute Decode

Signal	Function
LDCODEMEMREF HDCODEMEMREF	Decoded from op-code bits I2B00-05 and indicates the instructions which have a memory reference format used with load and store type instructions. Used to enable the Poll MAR logic.
LDCODEMEMREAD	Decoded from op-code bits I2B00-05. Used with opcodes that require a memory read. Generates read transaction codes. Read transaction code may be overlapped with higher priority transaction code such as Load Effective Address Real or Branch instructions. Also used to enable F & C bits for doubleword read operations.
HDCODEMEMWRT	Decoded from op-code bits I2B00-05. Used to indicate the opcode will do a memory write as either a first or second transaction (such as store word). Used as status flag for pipeline violation testing.
LDCODEZEROFBIT	Decoded from opcode bits I2B00-05. Causes the instruction or indirect word (F Bit) to be forced to zero in instructions where the F bit is an opcode bit (floating point instructions).
LDCODELEAR	Decoded from op-code bits I2B00-05. Used to generate a MAP transaction code. Overrides a read transaction for Load Effective Address Real Instrucitons.
HDCODELAORLEA LDCODELAORLEA	Decoded from op-code bits I2B00-05 and indicates opcodes 34 or D0 is non based mode or op-code 50(LA) in base mode. Used to load the effective MAR with effective address and arm the effective address interlock logic. Does not generate a CACHE transaction unless an indirect operation is encountered.
HDCODEINHDBLE	Decoded from opcode bits I2B00-05. Indicates that doubleword C-bits must be overridden. Used with Load Effective Address Real and Zero Memory Doubleword opcodes.
LSTORECKMAP	Decoded from opcode bits I2B00-05 and I2B00-08. Indicates that an instruction will do a write to memory. It requires a MAP access to predetermine if potential store errors (priveledged violation, map miss, or map invalid) exist. Used to cause a CACHE check map transaction code. Does not cause an actual CACHE or memory access.
LREADCHKSTORE	Decoded from opcode bits I2B00-05 and 12B-15. Indicates the opcode requires both a CACHE read and write. Causes a cache transaction code which indicates the cache must check for potential store errors during CACHE read operations.

continued

Table 3-7 (Cont.)
I2 Instruction Attribute Decode

Signal	Function
HLOCKLMAR	Decoded from op code bits I2B00-05 and 12B-15 or I2B00-08. Causes the logical MAR to be marked busy when the HLOCKLMAR signal enters I1. LMAR continues until a FLUSHPIPE, IRELEASELMAR, or BRANCH TAKEN order is given. Causes any subsequent instruction that uses LMAR to be frozen in I2 until the LMAR is released. It is used by instructions that: . put away to registers addressed by I bits 9, 10, 11 . put away to Base Registers . require LMAR or physical MAR during their execution steps . perform memory writes or operand perfetching that could potentially cause a conflict with the I/O instructions.
HMODREGTYPE	Decoded from opcode bits I2B00-05 and 12B-15. Indicates the instruction does a register put-away and a potential index register conflict exists.
HDMODRETYPE	Decoded from opcode bits I2B00-05 and 12B-15. Indicates double modify register type instructions. Marks immediate multiply and divide type instructions. If both modify register signals equal 1, the I2 register or lower contains an execute memory or execute register instruction.
HDCODEBRANCH LDCODEBRANCH	Decoded from op-code bits I2B00-08. Indicates branch instructions with memory reference formats where the branch target may be directly fetched. Causes an instruction read CACHE transaction code which overrides the read transaction codes. Inhibits Operand Request but sets Branch Request. Used with a FORCEWORD decode attribute.
LDCODEFRCWORD	Decoded from op-code bits I2B00-08. Indicates that the instruction F & C bits must be ignored for the CACHE transaction. Used with branch instructions.

store array (vector target). The uPC input multiplexer outputs an address selected from one of six sources. The uPC contains a full adder which selects either the direct data in, from the input multiplexer, the sequential address, which is the program counter address incremented by one, or the sub-routine return address from the control register. Figure F03-7 illustrates the uPC and the functions of each element. The 4-bit register is used to hold the CROM address during the CROM access cycle. The output from the uPC is the next macroinstruction address to the control store array.

The I1 register, loaded from the I2 register at the beginning of the CROM cycle, is used as a holding register for the instructions. The I1 register is automatically advanced to the I0 register at the end of the CROM cycle unless an I or E Unit wait exists. The CREG cycle then starts the execute stage.

3.6.4 Execute Stage (I0)

The final stage of the instruction pipeline is the execute stage. It is responsible for executing macro instructions, generating microinstructions, performing arithmetic/logic operations, and decoding microinterrupts. This information is now in the CREG cycle.

The I0 register automatically receives the instruction from the I1 register during this clock cycle. The I0 register holds the instruction during the execution phase and is advanced only under firmware control. If advanced, the I1/I0 look ahead multiplexer is enabled. This is also controlled by the firmware. During the CROM cycle, signal HADVANCEI1 is used as the select control line to the I1/I0 look-ahead multiplexer. HADVANCEI1 at 1 selects the I1 file address. HADVANCE I1 at 0 selects the I0 file address. The output of the MUX is dependent upon whether the operation is a read or write data to the internal registers of the 2901 or the dual-ported RAM. The A address multiplexer only selects read file addresses whereas the B address multiplexer selects both read and write file addresses. The actual register selected is dependent upon CROM bits 12-15 (for A register select) or CROM bits 16-19 (for B register select). The three bit fields (bits 6-8,9-11, and 13-15) output from the multiplexer were previously described in detail, in the I bus functions section of this chapter. The RAM register selection was previously described in the microprocessor operation portion of this chapter. Figure 3-16 shows the I1/I0 look ahead multiplexers, the register A and B multiplexer/registers, and their relationship with the 2901 microprocessor.

The control store array outputs the CROM information to the control register (CREG) and to the Register A and B multiplexer/registers. The CROM bits to the Register A and B multiplexer/registers, as shown in figure 3-16, are for the 2901 register selection. The CROM bits gated to the control register are transformed into CREG bits for properly sequencing microinstructions. The microword output from the CREG is gated directly into the 2901 via the CROM bus. The microword provides the basic control for the entire CPU. Chapter 4 is devoted to a description of the microword.

Arithmetic and/or logical operations are computed by the 2901. The 2901 is effectively the termination of the instruction pipeline. All outputs from the pipeline are gated to the 32 bit Y-Bus which distributes the data to both the CS unit and MS unit.

Once these four cycles have been completed, and after an instruciton has been executed, the cycles continue running repeatedly.

3.6.5 Advance Pipeline Routines

The pipeline is advanced by the firmware jump decode (JUMPD) sequence order. It is advanced without regard to the test conditions coded within the JUMPD. The specified test conditions (true or false) only controls the Micro Program Counter (uPC) input multiplexer. It also

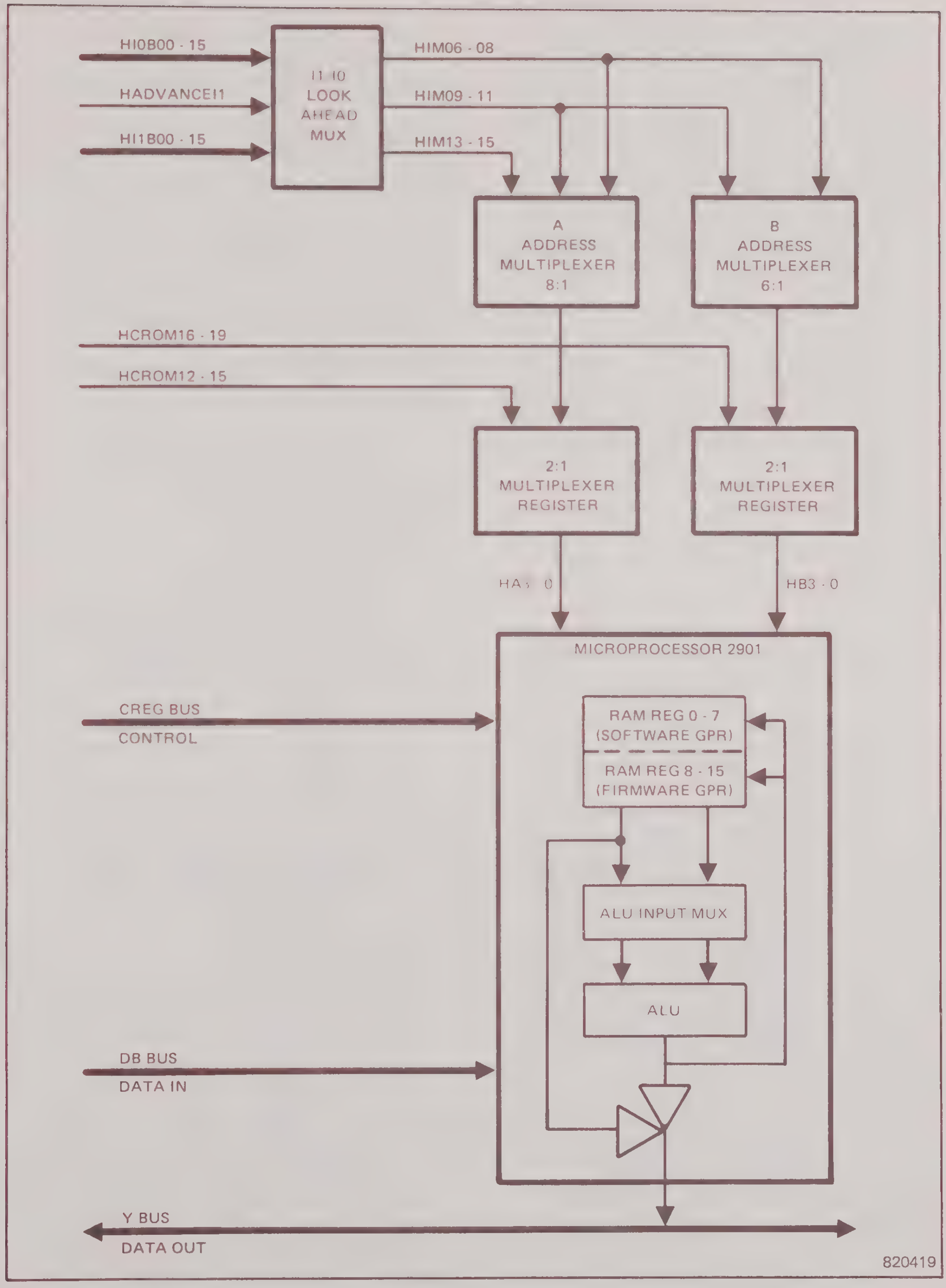

HI0B00 - 15
HADVANCEI1
HI1B00 - 15
I1 I0
LOOK
AHEAD
MUX
HIM06 - 08
HIM09 - 11
HIM13 - 15
A
ADDRESS
MULTIPLEXER
8:1
B
ADDRESS
MULTIPLEXER
6:1
HCROM16 - 19
HCROM12 - 15
2:1
MULTIPLEXER
REGISTER
2:1
MULTIPLEXER
REGISTER
HA3 - 0
HB3 - 0
MICROPROCESSOR 2901
CREG BUS
CONTROL
RAM REG 0 - 7
(SOFTWARE GPR)
RAM REG 8 - 15
(FIRMWARE GPR)
ALU INPUT MUX
ALU
DB BUS
DATA IN
Y BUS
DATA OUT
820419

Figure 3-16. 2901 Register Selection

determines if the decode vector is taken (test true) or uPC+1 (test false) is taken (refer to figure 3-8).

The JUMPD sequence order generates the BUMPPIPE signal on the MS unit. The BUMPPIPE signal is used to advance the pipeline. It is a CROM cycle signal which may be inhibited by either the CREG cycle of a microinterrupt return sequence or by a STOPCLOCK signal.

In the case of a microinterrupt return, the CROM cycle of the JUMPD (BUMPPIPE) is executed prior to the microinterrupt. Therefore, it must be inhibited during the microinterrupt return sequence in order to prevent a double BUMPPIPE order.

A STOPCLOCK inhibit of a BUMPPIPE signal can happen under varing circumstances; however, the two primary conditions for a STOPCLOCK inhibit signal are:

> The instruction in execution causes a microinterrupt.
>
> The instruction in I1 required a CACHE access and did not gain access to the CACHE (CACHE miss).

The BUMPPIPE signal is a CROM cycle that is generated by the microinstruction which represents the macroinstruction in I1. However, the macroinstruciton in I1 must be executable by one microinstruction. If a macroinstruciton rquires two or more microinstructions for execution, the macroinstruction resides in I0 before the BUMPPIPE signal is generated. In this case, I1 is invalid until the next time that the pipeline is advanced.

Although the BUMPPIPE signal is used to qualify the clocks for the pipeline registers, it functions in both the CROM and CREG cycles as follows:

> CROM Cycle: The entire pipeline is advanced assuming the rules for advancement are met.
>
> CREG Cycle: The I0 register is advanced a second time. If I1 is invalid, then I0 is invalidated during the second advance.

The general rules for advancing the pipeline are as follows:

I3 is advanced if:
- a) I2 is empty.
- b) I2 is full but is advanced to I1 during the advance.

I3 is loaded with a new instruction from cache or invalidated if the cache output is invalid. I3 is loaded with a valid instruction if:

- a) I3 is empty.
- b) I3 is full but is advanced to I2 during the advance.

I3 is not loaded with a new instruction or is invalidated if:

- a) I3 contains a right halfword instruction that has not been advanced (or will not be advanced) to I2 during this advance.
- b) I3 is full and I2 if full and cannot be advanced during this advance.
- c) I3 contains a Branch Target instruction and the associated branch instruction has not been committed. In this case I2 will be invalidated if advanced.

I2 is advanced if:

- a) The instruction in I2 requires a logical MAR access and if LMAR access was achieved.
- b) The instruction in I2 requires a logical and effective MAR access and the access was achieved.
- c) All direct addressing related to the instruction in I2 has been resolved.
- d) The instruction in I2 does not have an index register conflict with either the instruction in I1 or I0.

If I2 can not be advanced or if I2 is invalid, then I1 will be set invalid.

I1 advances automatically one clock after it is set valid unless a STOPCLOCK condition is present. The automatic advance is used since the contents of I1 represents a micro instruction CROM cycle which can only be true for one clock cycle. The STOPCLOCK condition only effects the I2, I1, and I0 levels of the pipeline. The stop clock also indicates either a microinterrupt of the instruction in I0 or the lack of a required cache access by the instruction in I1.

I1 can be advanced by either:

a) The automatic advance of a valid instruction from I1 to I0.
b) The CREG cycle of a JUMPD (BUMPPIPE) sequence.

If I1 is invalid during an advance sequence, then I0 will be invalidated by that advance.

3.6.6 Pipeline Conflict Logic

The purpose of the piepline conflict logic is to detect conflicting instruction conditions in the pipeline. An example of a conflict is when a prefetched instruction is currently resident in the pipeline, but is due to be modified by one of the instructions that precede it. The conflict is detected by comparing the LMAR and the I3PC (N+2 LMAR) address register contents.

Pipeline conflicts occur most commonly with store instructions. When an instruction that modifies memory is decoded, and an address match is detected, a conflict exists. In this case, the IE Unit does not use the instruction currently in I3PC, but refetches the instruction using the same address.

3.6.7 Pipeline Violation Logic

The pipeline violation logic checks the status of the pipeline for a specific instruction sequence. If a store instruction, which modifies memory, is in the execute stage of the pipeline, the remaining pipeline registers may contain invalid data. With this type of violation, the pipeline is flushed and refilled with the valid data that the store instruction (executed in I0) put into memory. The sequence of events is monitored and executed by the pipeline violation PROM.

The pipeline violation PROM checks status flags relating to instructions contained in the pipeline. If pipeline register I0 contains a store instruction, and that instruction is executed, the PROM determines whether the remaining instructions in the pipeline are still valid and whether or not they are halfwords. This is accomplished by comparing the contents of the LMAR with the contents of the I3PC register (CS Unit). If the comparison is good, the comparator outputs an address equal (HADDREQUAL) signal, which is sent to the violation PROM. The PROM simultaneously checks the store instruction in I0 against the contents of the I2 and I3 registers. If a match occurs, a pipeline violation is evident.

In the event of a pipeline violation, a flip-flop is set to output a refill pipeline (HI2REFILLPIPE) signal. This signal is routed to the MS Unit and a decode exception is generated. This causes the vector logic to stop generating true vectors and to generate exception vectors. The instruction in I3 is simultaneously clocked into I2; however, the corresponding status flag is not set which invalidates the instruction. When this instruction moves into the execute (I0) stage, the firmware reads the refill pipeline signal, the prefetched data is flushed from the pipeline, and the pipeline is refilled with the new data put into memory by the store instruction.

An exception to the sequence occurs when I0 contains a store instruction and I1 contains a branch instruction. In this case, a pipeline violation exists but the branch flushes the violation.

3.6.8 Address Specification Errors

An address specification error occurs when a particular instruction violates address specification rules applicable to that instruction. Table 2-14 lists the instruction operation codes together with their attributes and refers to the rules applicable for op codes by number. The conditions (rules) that generate address specification errors are listed in a separate table (table 2-15).

The vectors and rule terms for an address are generated in the MS Unit when the instruction is at the precode (I2) stage. Up to two variations of the rule terms (0, 1 and 2) are also generated depending upon which sub-decode PROMs are being used. The output from the two PROMs is routed through a 2:1 multiplexer/latch which effectively acts as an I1 stage holding latch. The rules governing the address are clocked into the I1 pipeline register together with the instruction. The two rule PROMs on the IE Unit decipher the address specification rules plus the various signal conditions of the pipeline contained in I1. This determines whether one of the address specification rules is applicable. If it is, an HI1ASERROR signal, an HI0ASERROR signal, and the instruction with which the error is associated are clocked into I0. This generates a microinterrupt vector signal which is routed back to the MS Unit and used to generate a microinterrupt.

3.6.9 Auto Microinterrupt

This feature is set automatically when the system is reset by enabling the enable auto microinterrupt level order. Once enabled, it remains set until the system is powered down unless micro diagnostics are being run. In this case, the auto microinterrupt feature is disabled.

The auto microinterrupt is only effective as a valid instruction goes into the I0 stage, and only on the first cycle of a multicycle instruction. If an instruction, which is being decoded and routed into I0, contains a decode exception, an auto microinterrupt does not occur since the decode exception has priority. Conversely, if the instruction does not contain a decode exception, the flip-flop is enabled for one cycle as the instruction goes into I0 and is then disabled.

The auto microinterrupt logic monitors the I0 stage for instructions that contain a microinterrupt. The logic then causes the microinterrupt to be accepted. If an instruction contains nested microinterrupts, the logic handles all of them until the instruction becomes executable. This is accomplished by generating a return signal (HICRORTNTL) from the microinterrupt to the microinterrupt logic, thereby reenabling the logic for an additional cycle.

3.7 Cache

The cache is loaded at medium speed from SelBUS main memory. It is then accessed at high speed. This technique, employing the fast access cache memory, is used to attain higher operating speeds. Figure 3-17 is a block diagram of the relevant hardware.

3.7.1 Introduction

At any time, the cache can only contain a copy of portions of the much larger main memory. The actual data written into the cache is determined by the assumption of certain basic principles. The principle of program locality infers that programs have a tendency to make most accesses in the region of locations accessed in the recent past. Programs typically execute instructions in straight lines or small loops, with the next few accesses likely to be a few words ahead of or behind the current instruction. Logic elements such as stacks grow and shrink from one end, with the next few accesses near the current top. Also, hardware data elements are often scanned sequentially. The cache uses these behavioral characteristics by bringing in extra words on each access to main memory (look ahead) and keeping copies of recently used words (look back).

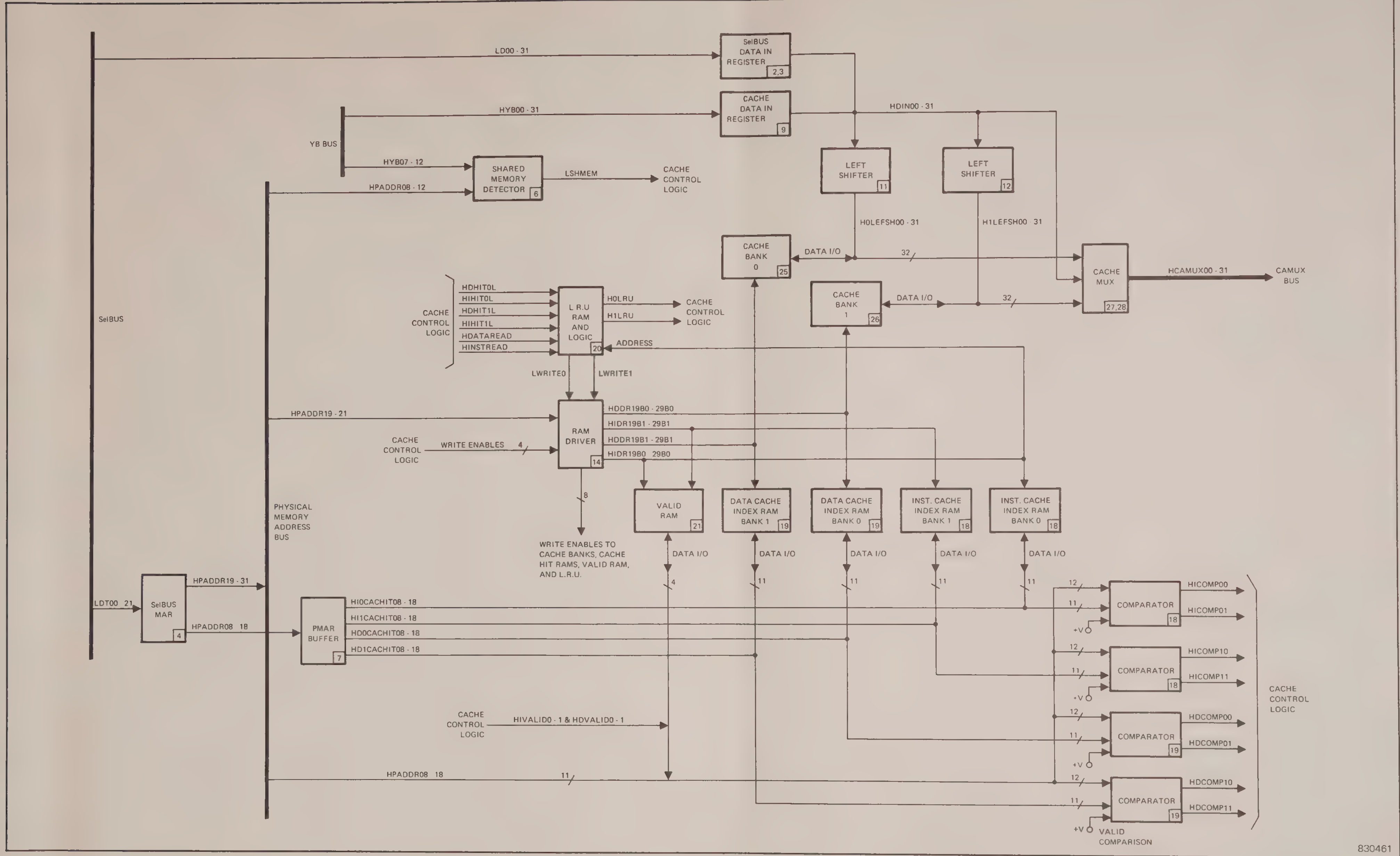

SelBUS
LD00 - 31
SelBUS DATA IN REGISTER 2,3
YB BUS
HYB00 - 31
CACHE DATA IN REGISTER 9
HDIN00 - 31
HYB07 - 12
HPADDR08 - 12
SHARED MEMORY DETECTOR 6
LSHMEM
CACHE CONTROL LOGIC
LEFT SHIFTER 11
LEFT SHIFTER 12
H0LEFSH00 - 31
H1LEFSH00 31
CACHE BANK 0 25
DATA I/O
32
CACHE BANK 1 26
CACHE MUX 27,28
HCAMUX00 - 31
CAMUX BUS
CACHE CONTROL LOGIC
HDHIT0L
HIHIT0L
HDHIT1L
HIHIT1L
HDATAREAD
HINSTREAD
L.R.U RAM AND LOGIC 20
H0LRU
H1LRU
CACHE CONTROL LOGIC
ADDRESS
LWRITE0
LWRITE1
HPADDR19 - 21
RAM DRIVER 14
CACHE CONTROL LOGIC
WRITE ENABLES 4
HDDR19B0 - 29B0
HIDR19B1 - 29B1
HDDR19B1 - 29B1
HIDR19B0 29B0
8
WRITE ENABLES TO CACHE BANKS, CACHE HIT RAMS, VALID RAM, AND L.R.U.
VALID RAM 21
DATA CACHE INDEX RAM BANK 1 19
DATA CACHE INDEX RAM BANK 0 19
INST. CACHE INDEX RAM BANK 1 18
INST. CACHE INDEX RAM BANK 0 18
DATA I/O
4
11
PHYSICAL MEMORY ADDRESS BUS
LDT00 21
SelBUS MAR 4
HPADDR19 - 31
HPADDR08 18
PMAR BUFFER 7
HI0CACHIT08 - 18
HI1CACHIT08 - 18
HD0CACHIT08 - 18
HD1CACHIT08 - 18
CACHE CONTROL LOGIC
HIVALID0 - 1 & HDVALID0 - 1
HPADDR08 18
11
12
COMPARATOR 18
HICOMP00
HICOMP01
+V
COMPARATOR 18
HICOMP10
HICOMP11
COMPARATOR 19
HDCOMP00
HDCOMP01
COMPARATOR 19
HDCOMP10
HDCOMP11
VALID COMPARISON
CACHE CONTROL LOGIC
830461

Figure 3-17. Cache Block Diagram

Programs tend to contain common blocks of data (operands), as opposed to intermixing instructions and operands. Advantage is taken of this tendency by providing separate instruction and operand caches. This also ensures that data written into the cache will not overwrite instructions.

Since the cache contains only portions of main memory, its effectiveness is based on the percentage of time that the required data can be fetched from the high-speed cache rather than the lower speed main memory. A cache HIT occurs if the data is found in the cache. A cache MISS occurs if the data is not found in the cache and has to be fetched from main memory.

3.7.2 Primary Organization

The cache memory comprises 16 4K by 4-bit RAM ICs configured to provide an 8K by 32-bit word cache. The total cache is divided into two 4K banks, designated bank 0 and bank 1. Each of the 4K banks is further subdivided into a 2K operand cache and a 2K instruction cache. Each location contains one 32-bit memory word, excluding parity or error correction bits.

With one 2K-word operand or instruction cache and a 4M-word main memory, there are 2048 possible memory words that can be stored in any given location. However, there are two 2K-word operand caches and two 2K-word instruction caches. In the worst case of a 4M-word program consisting entirely of instructions, only two of the 2048 possible alternatives can be cache resident at any time. A cache index array is provided to track the storage of words in the cache locations. The index array comprises four 4K by 12-bit RAM ICs. Two index RAMs are assigned to cache bank 0 and two are assigned to cache bank 1. In each case, one index RAM is associated with instructions and the other with operands (data). Each index RAM location contains the 11 most significant bits of a physical word address (only 11 of the 12 bits in each location are used).

A valid RAM is used to signify whether locations in the cache banks contain valid information. The valid RAM comprises two 4K by 4-bit RAM ICs. Each single bit in the valid RAM corresponds to a cache location (data and instruction locations in both cache bank 0 and cache bank 1). Therefore, every location in the cache can be marked valid or invalid.

3.7.3 Cache Addressing

The cache banks, the index RAMs, and the valid RAM are addressed in parallel by a 22-bit physical word address (HPADDR08 thru 29) from the physical memory address bus. Address bits HPADDR19 thru 29 are routed via the RAM driver to select the same location in all seven RAMs (two cache, four index, and the valid RAM). Address bits HPADDR08 thru 18 are routed around the RAMs to four comparators to be compared with the output from each index RAM.

3.7.4 Write to Cache

Writes to cache are executed either via the SelBUS data in register or via the cache data in register. Data is routed in from the SelBUS over the LD00 thru 31 lines into the SelBUS data in register during data returns from main memory. Data is routed from the YB bus over the HYB00 thru 31 lines into the cache data in register during CPU to cache writes. Both of these registers transfer data onto a common 32-bit bus (HDIN00 thru 31). Depending upon whether the data is to be written into cache bank 0 or cache bank 1, one of the two left shifters aligns the data to byte, halfword, or word boundaries. The appropriate cache bank is then loaded via the H0LEFSH00 thru 31 lines or the H1LEFSH00 thru 31 lines.

To increase the speed of the cache fill-up sequence, the control logic reads a doubleword pair from SelBUS main memory and stores the doubleword in the cache. The cache logic always reads doublewords even though the cache access may have been for a byte or a halfword. The doubleword is fetched using back-to-back memory read transfers (MRTs) to the even and odd memory words. An exception is if the CPU accesses a memory location in a shared memory block. In this case, the control logic only reads one word.

3.7.5 Least Recently Used (L.R.U.) Circuitry

Both cache banks are addressed in parallel and valid data can exist in the same location in both banks. When it is required to load data into the cache under these circumstances, a decision is made to determine into which bank the new data will be loaded. This decision is based upon which cache bank was last used (actually, which was least recently used). A RAM in the L.R.U. circuitry is always addressed in parallel with the cache banks, index RAMs, and valid RAMs. The 12-bit address to the 4K by 4-bit L.R.U. RAM comprises the HIDR19B0 thru 29B0 lines and a signal which indicates whether a cache instruction bank or a cache data bank has been accessed. Two of the four data I/O lines from the RAM are used as flags (HOLRU and H1LRU) to indicate which cache bank has been used least recently for any addressed location (one flag is set, the other is reset). These indications are always provided, but they are only used if a cache write conflict exists. If a conflict exists, the flags are sampled to detect which cache bank was used least recently. The data is then written into this bank, since the bank most recently used is likely to contain more relevant information which should be retained. A write enabled signal (LWRITE0 or LWRITE1) is issued to the appropriate cache bank via the RAM Driver.

3.7.6 Externally Originated Writes

The cache must contain a reliable copy of SelBUS main memory. Therefore, each externally originated memory write must be monitored (byte, halfword, or fullword). This is the primary function of the SelBUS interface. The SelBUS memory address of each externally originated memory write is captured and a cache access is initiated. A cache hit signifies that information exists in the cache under that address. However, this data is now incorrect because new data has been written to main memory under that address. The cache resident data must, therefore, be invalidated. This is accomplished by resetting the valid bit in the corresponding valid RAM location. The data that is now invalid remains inaccessible in the cache until it is overwritten with valid data.

3.7.7 CPU Write to Cache

In order to avoid discrepancies between the cache and main memory, all CPU writes to cache must also be written to main memory. The memory address is captured at the SelBUS interface and the information is written over the SelBUS to that location in main memory.

3.7.8 Read From Cache

In order to determine if a specific memory location is cache resident, the seven RAMs are addressed as previously described. If one of the outputs from the comparators signifies equal comparison on the HICOMP or HDCOMP lines, the specific memory location is cache resident. This can only occur if the information in the cache is marked valid in the valid RAM. The addressed output bit from the valid RAM is applied to the comparators together with the HPADDR08 thru 18 lines. It is compared with a permanent high logic level forming the twelfth bit on the other input to each comparator (11-bit output from the index RAM and permanent high at comparator). Determination of whether the word is in cache bank 0 or cache bank 1 is inherently made by the index RAM that generated the equal comparison.

If no equal comparison is found, the specified memory location is not cache resident. This is decoded as a cache miss by the cache control logic, and the required information is fetched from main memory.

During normal operation, starting from a reset state, the bits in the valid RAM are reset. When the CPU begins to execute instructions, an attempt is made to read operands and instructions from the cache. Since the bits in the valid RAM are reset, a cache miss occurs. The control logic takes over to fetch the required information from SelBUS main memory using SelBUS memory read transfers (MRTs). The requested information is returned using SelBUS data return transfers (DRTs). Each DRT received is stored in the cache. The 22-bit physical address is routed in on the LDT00 thru 21 lines. At the SelBUS MAR, the HPADDR19 thru 31 lines are output onto the physical memory address bus to address the cache banks, index RAMs, and valid RAM via the RAM driver. The HPADDR08 thru 18 bits are routed via the PMAR buffer and stored in the corresponding location in the index RAMs. Also, providing there are no errors (parity, etc.), appropriate bits in the valid RAM are set.

Handling of the information fetched from main memory varies according to whether it is an instruction or an operand. The requirement for an operand fetch is detected at the I2 stage of the pipeline (decode stage). If the operand is not found in the cache, it must be fetched from main memory before the instruction can be executed. The instruction is advanced to the I1 (CROM stage). The I1 and I0 (execution stage) are then stopped until the operand is fetched. When the data is returned, it is written into the cache and simultaneously loaded into the cache data out register on the MS unit (via the cache multiplexer and the CAMUX bus) for use by the instruction in the pipeline. Operand alignment is performed (if required) by right shifters on the MS unit before the data is loaded into the cache data out register.

An instruction fetch varies in two ways depending upon whether execution of the instruction in the I0 stage of the pipeline is finished. If execution of the instruction is not finished, the pipeline is still full. In this case, the returned data is written to the cache. When the instruction in I0 has been executed, the pipeline flows, and the I3 register (predecode stage) is empty. The cache is then accessed and the instruction is loaded into I3.

If the instruction in the I0 register has been executed by the time the new instruction has been fetched from main memory, the I3 register is already empty. The new instruction is written into the cache and simultaneously loaded into I3.

3.7.9 Cache Memory Control Register

The cache memory control (CMC) instruction is used to clear and to turn on/off the cache banks. The contents of a specified general purpose register are loaded into the cache memory control register on the CS Unit via the Y Bus (YB23 through YB31). The bit pattern then determines how the cache is controlled. If set, bits 23 through 26 clear instruction cache bank 0, instruction cache bank 1, operand cache bank 0, and operand cache bank 1 respectively. If set, bits 27 through 30 enable (turn on) the cache banks in the same respective order (bit 27 instruction cache bank 0, bit 30 operand cache bank 1). If bits 27 through 30 are reset, the corresponding cache banks are disabled (turned off). Bit 31 is normally used for diagnostic purposes. If bit 31 is reset, the instruction cache banks are bypassed.

3.7.10 Cache Shifter (F and C Bits)

The F and C bits in a memory reference instruction control the cache shifters. The F bit determines whether the operation is a word or byte type. The C bits determine the shift. The F and C bits are routed from the SelBUS and used to address a decode PROM as signals HFBIT and HCBIT30/31 respectively. The output from the PROM controls the related shift logic on the MS Unit. The data type (doubleword, word, halfword, or byte) and the shifts for combinations of F and C bits are as follows:

F BIT 12	C BITS 30	31	DATA TYPE	SHIFT
0	0	0	32-bit word	No shift
0	0	1	Left halfword	Shift bits 00-15 to 16-31
0	1	0	64-bit doubleword	No shift
0	1	1	Right halfword	No shift
1	0	0	Byte 0	Shift bits 00-07 to 24-32
1	0	1	Byte 1	Shift bits 08-15 to 24-31
1	1	0	Byte 2	Shift bits 16-23 to 24-31
1	1	1	Byte 3	No shift

Sign extension is required for both left and right halfword data types. For a left halfword, bit 16 is examined after the shift has been done. If bit 16 is a logical 0, bits 00 through 15 are filled with 0s (i.e., bytes 0 and 1 = 00 hex). If bit 16 is logical 1, bits 00 through 15 are filled with all 1s (i.e., bytes 0 and 1 = FF hex). A right halfword is sign extended in the same way, but there is no prior shift. Byte data types are always right justified. Therefore, bit 00 through 23 are zero extended (i.e., bytes 0, 1, and 2 = 000 hex).

3.7.11 Shared Memory Detection

In a system with two CPUs, the area of memory from 128K up to the maximum of 4M words may be shared. In this case, the external path to the shared memory is by way of alternate ports. Therefore, one CPU cannot monitor memory transactions that occur through the other port. Unless 100 percent of memory transactions can be monitored, the cache cannot be kept up to date.

The actual area of memory to be shared is defined by software in the form of an upper and a lower limit in blocks of 128K words. This information is routed in on the YB bus and stored in a register within the shared memory detector circuitry. Accesses to main memory are constantly monitored and compared with the information stored in the limit register. If an access to the area of shared memory is detected, the cache is bypassed and the read or write is executed directly to or from main memory.

In a system two CPUs, if one CPU generates a set bit in memory (SBM) or zero bit in memory (ZBM) instruction, a read and lock (R&L) is generated within the specified shared memory boundaries. The other CPU is then prevented from performing a SBM or a ZBM instruction until the original instruction is complete. However, the second CPU can perform other memory transactions (memory read, memory write) during the time that the R&L is active.

In a system with a CPU/IPU configuration, which share the same private memory, all of the memory is locked out until the instruction is completed.

3.7.12 Instruction/Operand Stop

Functions such as break points may be implemented at the CRT panel. These include instruction stop (IS), operand read stop (RS), and operand write stop (WS). Two sets of panel attention and I/O ready lines from the SelBUS are monitored (LSCPTTN and LREADY for the CPU, LIPPTN and LIPREADY for the IPU). These lines are used to sense and control address stops. The mode, CPU or IPU, is detected at the MS Unit and signal HIPU is used to select the appropriate lines via a multiplexer. The selected ready line is routed via SelBUS tag logic to produce an

HRDY flag which is examined by the firmware. The corresponding panel attention line is routed via error status logic to produce either a qualify address stop (HQULADST) signal for an instruction stop or an operand address stop (HADDRSTOPERR) signal. In the case of the IS and RS functions, the cache is turned off and the processor reverts to the single-word fetch mode (as opposed to double-word fetches from cache).

3.8 Writable Control Store (WCS)

Entry and use of the WCS, together with some basic user notes, are provided in the following paragraphs. Detailed software control and initialization information is contained in Chapter 2.

3.8.1 Entry and Use

Entry into WCS from software is accomplished using the JUMP WCS macroassembler instruction. This instruction allows the user to jump to any of the 4K or 8K locations within WCS (which were initialized when WCS was loaded) where vector addresses (in microcode) are stored which address routines within the WCS. When calling any of the WCS user written routines from software, the user accesses these routines through JUMP WCS.

The writing of WCS is accomplished using the WRITE WCS macroinstruction. The reading of WCS is accomplished using the READ WCS macroinstruction. The protocol and use of the WCS instructions are fully covered in the Gould CONCEPT 32/67 Reference Manual.

The use of each WCS instruction is briefly described in the following list.

1. Jump WCS instruction. To allow the user access to the WCS.
2. Write WCS instruction. Allows the user the ability to write to the WCS.
3. Read WCS instruction. Allows the user to read the contents of WCS. This allows the user to verify the contents of the WCS against the data that was loaded into the WCS, and accommodates storage and retrieval of data in WCS. The user may desire to use WCS for temporary storage of program data. This instruction allows retrieval of that data.
4. Set CPU instruction. Used to switch the CPU from operating under the main firmware set (PROM) to ACS. RD(bits 6, 7 & 8) of the instruction will specify the GPR in the CPU that will define the operating characteristics of the CPU. If bit 21=1, the CPU will operate under ACS control. If bit 21=0, the CPU will operate under CROM control (default).
5. Read CPU status instruction places the current operational status of the CPU into the R_D register.

3.8.2 WCS User Notes

When utilizing the WCS, the following points should be noted.

1. If there is no keyword in the scratchpad, system Reset initializes all locations in WCS, requiring the user to reload.
2. Power failure destroys the contents of WCS, necessitating that the user reload all of the WCS.
3. The JWCS instruction is addressed in the range of 1000-2FFF.
4. All executable code contained in WCS is written in microcode. Standard assembler coding is not executable by WCS.

3.9 Memory Map

The memory management hardware permits full utilization of all available memory and includes memory allocation and protection (MAP). Figure 3-18 is a block diagram of the relevant memory management hardware.

3.9.1 Introduction

Any user's task requires a certain amount of storage space (logical space), and sequential software-generated logical addresses are assigned to the program. However, in order to make the most efficient use of available memory in a multiprogram environment, programs are not necessarily stored sequentially within the physical memory. Therefore, for a program to run, the logical addresses must be converted into physical addresses. As a basic definition, mapping is the mode of operation in which software-generated logical addresses are transformed through the memory map so that they become references to actual physical locations in main memory. Different users may have access to some of the same physical address space and occasionally share these common blocks of memory.

A program may be dispersed into non-contiguous 2K-word blocks throughout the physical memory. The map image of a given task tracks this dispersion. All map images are controlled by the software operating system. When the CPU is operating in the mapped environment, firmware monitors all programs that are currently running and appropriate parts of the map images are brought in as required. The techniques used to clear a map image, load another, and access it are fast enough to ensure that program appear to run concurrently.

Normally, the map contains the required data, the logical to physical address conversion is performed, and the cache or main memory is accessed. However, under certain circumstances, registers in the map may not be loaded or they may be loaded with invalid data. An indication of either of these conditions is provided. If an empty or invalid map location is addressed, a map miss signal is generated and routed to the IE unit as a microinterrupt. A sequence is initiated to fetch the required map entry from cache or main memory, validate it, and load it into the map. The map is then reaccessed. During this time, the macroinstruction pipeline is frozen if the miss occurred on an operand fetch; it is flushed and refilled if the miss occurred on an instruction fetch.

3.9.2 Memory Mapping Data Structures

Figure 3-19 depicts the software memory mapping data structures used by the CPU to load its map. The master process list (MPL) and the map image descriptor list (MIDL) must be kept in memory on doubleword boundaries. These contain the information required by the CPU to load the map. MPL0 is normally reserved for the operating system (OS). The remaining MPLs are used for tasks (programs) within the OS.

3.9.2.1 Current Process Index

The second word of the program status doubleword (PSD) contains the current process index (CPIX) field. The 14-bit CPIX field provides the index that is used to locate the map segment descriptors (MSDs) in the MPL, thereby providing a link form the PSD to the map image descriptors (MIDs). The CPIX must point to a doubleword boundary; therefore, the three least significant bits of the 14-bit field are always zero.

3.9.2.2 Master Process List

The map segment descriptors (MSDs) are contained in the MPL. The address of the MPL is set at system reset time by loading a predetermined scratchpad cell (F3 hex) with the 24-bit physical MPL address. This location points to MSD0. Therefore, when the CPIX = 0, the MIDs for MSD0 are used. If the CPIX is not equal to zero,

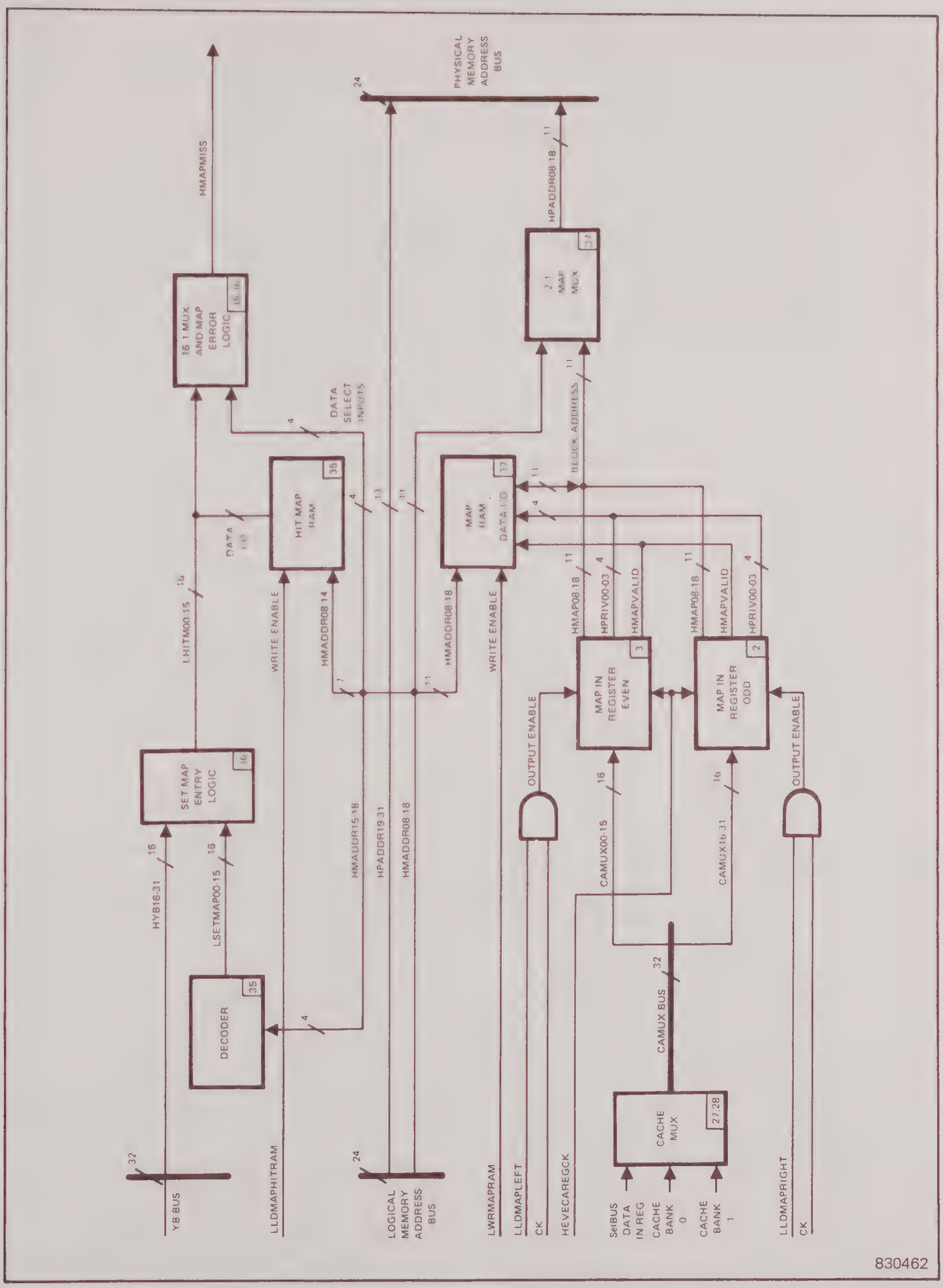

YB BUS
32
LLDMAPHITRAM
LOGICAL MEMORY ADDRESS BUS
24
LWRMAPRAM
LLDMAPLEFT
CK
HEVECAREGCK
SelBUS DATA IN REG
CACHE BANK 0
CACHE BANK 1
LLDMAPRIGHT
CK
DECODER
35
4
HYB16-31
16
LSETMAP00-15
16
SET MAP ENTRY LOGIC
36
LHITM00-15
16
DATA I/O
HIT MAP RAM
35
WRITE ENABLE
HMADDR08-14
7
HMADDR15-18
HPADDR19-31
HMADDR08-18
11
4
13
11
DATA SELECT INPUTS
4
16:1 MUX AND MAP ERROR LOGIC
15, 16
HMAPMISS
MAP RAM DATA I/O
37
WRITE ENABLE
HMADDR08-18
OUTPUT ENABLE
MAP IN REGISTER EVEN
3
MAP IN REGISTER ODD
2
OUTPUT ENABLE
CAMUX00-15
16
CAMUX16-31
16
CAMUX BUS
32
CACHE MUX
27/28
HMAP08-18
11
HPRIV00-03
4
HMAPVALID
HMAP08-18
11
HMAPVALID
HPRIV00-03
4
11
4
BLOCK ADDRESS
11
2:1 MAP MUX
37
HPADDR08-18
11
24
PHYSICAL MEMORY ADDRESS BUS
830462

Figure 3-18. Memory Management Hardware Block Diagram

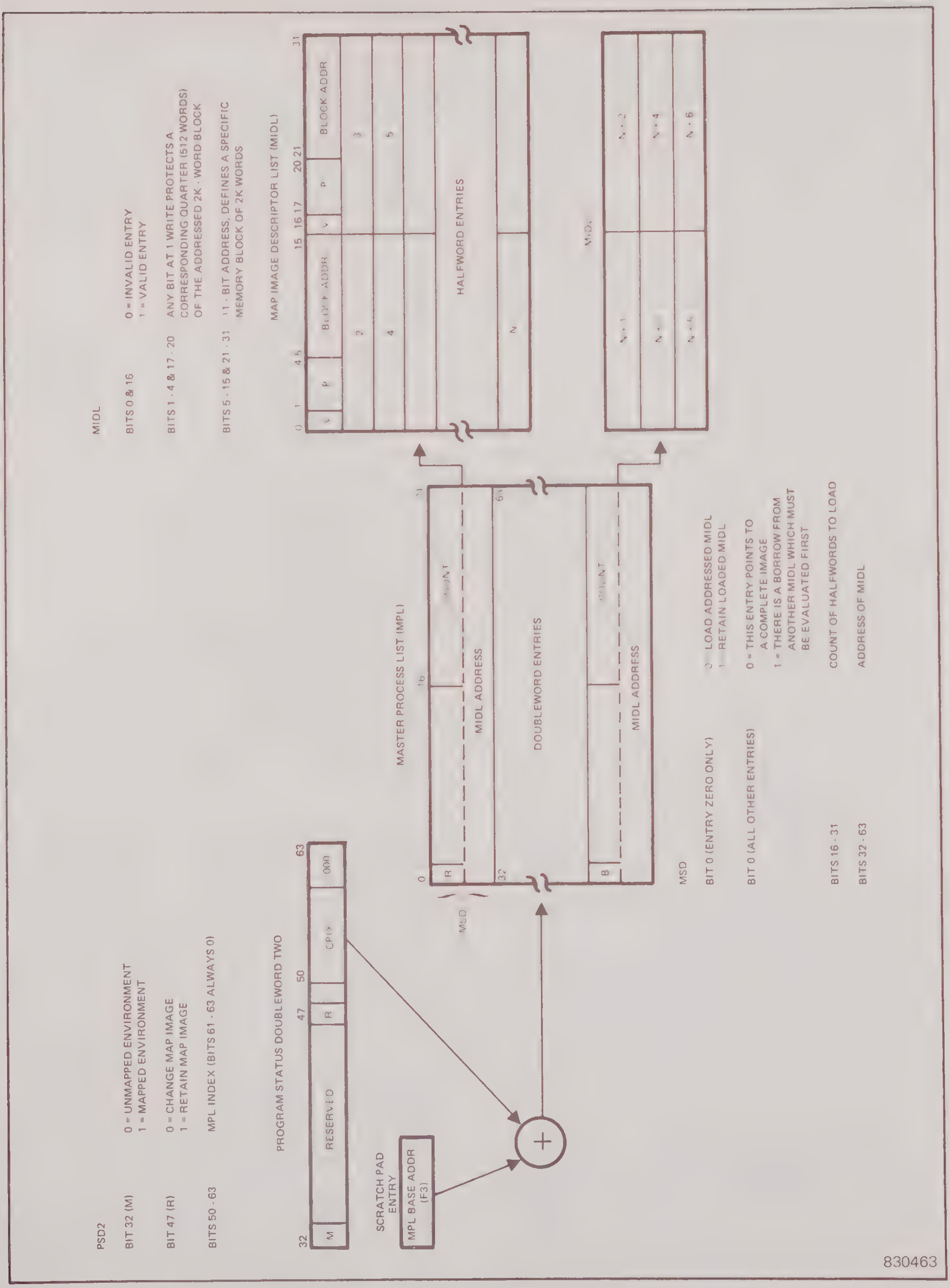

PSD2
BIT 32 (M)
0 = UNMAPPED ENVIRONMENT
1 = MAPPED ENVIRONMENT
BIT 47 (R)
0 = CHANGE MAP IMAGE
1 = RETAIN MAP IMAGE
BITS 50 - 63
MPL INDEX (BITS 61 - 63 ALWAYS 0)
PROGRAM STATUS DOUBLEWORD TWO
32
47
50
63
M
RESERVED
R
CPIX
000
SCRATCH PAD ENTRY
MPL BASE ADDR (F3)
MASTER PROCESS LIST (MPL)
0
16
31
32
63
R
COUNT
MIDL ADDRESS
DOUBLEWORD ENTRIES
B
COUNT
MIDL ADDRESS
MSD
MSD
BIT 0 (ENTRY ZERO ONLY)
0 = LOAD ADDRESSED MIDL
1 = RETAIN LOADED MIDL
BIT 0 (ALL OTHER ENTRIES)
0 = THIS ENTRY POINTS TO A COMPLETE IMAGE
1 = THERE IS A BORROW FROM ANOTHER MIDL WHICH MUST BE EVALUATED FIRST
BITS 16 - 31
COUNT OF HALFWORDS TO LOAD
BITS 32 - 63
ADDRESS OF MIDL
MIDL
BITS 0 & 16
0 = INVALID ENTRY
1 = VALID ENTRY
BITS 1 - 4 & 17 - 20
ANY BIT AT 1 WRITE PROTECTS A CORRESPONDING QUARTER (512 WORDS) OF THE ADDRESSED 2K - WORD BLOCK
BITS 5 - 15 & 21 - 31
11 - BIT ADDRESS, DEFINES A SPECIFIC MEMORY BLOCK OF 2K WORDS
MAP IMAGE DESCRIPTOR LIST (MIDL)
0
1
4
5
15
16
17
20
21
31
V
P
BLOCK ADDR
V
P
BLOCK ADDR
2
3
4
5
HALFWORD ENTRIES
N
MIDL
N + 1
N + 2
N + 3
N + 4
N + 5
N + 6
830463

Figure 3-19. Mapping Data Structures

the CPIX and location F3 are added together. The result of this addition points to the MSD entry other than MSD0 (on a doubleword boundary).

In the format of an MSD entry, bit 0 of word 0 is interpreted one way for MSD0 and another way for all other MSDs. For MSD0, bit 0 is considered as the retain bit (R). For any other MSD, bit 0 is considered as the include bit (I). A load program status doubleword and change map (LPSDCM) instruction for a context switch require that the map be changed. The firmware then determines the appropriate MSD to retrieve by adding the CPIX portion of PSD2 to the MPL base address located in the scratchpad (location F3). When this MSD is retrieved, the firmware analyses bit 0 (I bit). If bit 0 is equal to zero, all maps described by the CPIX are used. The hit RAM is zeroed and the look-aside buffer pointer points to the CPIX MSD.

If bit 0 of the retrieved MSD is equal to one, the firmware examines MSD0. If bit 0 (R bit) is equal to zero, an absolute load of all maps described by MSD0 occurs and the CPIX offset is computed. This occurs once during system initialization and after changes to the MSD0 map block.

Bit 0 of MSD0 equal to one signifies that the map blocks of MSD0 must be retained and the computed CPIX offset must be used. Therefore, the executing task uses the map blocks defined by the computed CPIX offset MSD to translate the logical address of the instruction or operand into a physical memory address while retaining the map blocks of the OS (MSD0).

The only time that bit 0 of MSD0 = 0 and bit 0 of the computed CPIX MSD = 1 is during offline diagnostic testing and once after each reset.

A fault condition occurs if the CPIX = 0 and bit 0 of MSD0 = 1.

Bits 1 through 15 of word 0 in the MSD format are reserved for future use. Bits 16 through 31 are the segment count which signifies the number of map block entries in the MIDL.

Word 1 of the MSD contains the MIDL pointer. This is the physical address of the first map image descriptor (MID) in the MIDL. The MIDL must point to a word address (bits 30 and 31 are set to zero).

3.9.2.3 Map Image Descriptor List

The MIDL maps logical addresses into physical addresses. Each MIDL entry associates a map block of the logical address space with a map block of physical memory.

3.9.2.4 Map Image Descriptor

Each MID is a halfword that contains an 11-bit map block entry, a map block valid/invalid bit, and four write protect/unprotect bits. By selecting one of the four write protect bits, any quarter of any map block may be designated as write protected. The map blocks contained within the MID may or may not contain valid information for address translation. The look-aside buffer (described later) is used to determine if the map block in the reference MIDL is valid. If it is valid, a map hit occurs and logical to physical address translation is performed. If it is invalid, a map miss occurs and the firmware executes a memory reference transaction to load the selected MID with a valid map block. The logical to physical address translation is then performed.

3.9.2.5 Map Initialization

When a new PSD is being enered into the CPU, there are three possible CPU actions relating to the map.

When the unmapped mode is set, the CPU deactivates the map for the duration of the execution of this PSD. (An unmapped indication in the PSD overrides the LPSDCM instruction.)

When the LPSD instruction is used to load the PSD and the mapped mode is set, the CPU activates the map circuitry and uses whatever is in the map.

Except for the two preceding cases, entry of a new PSD into the CPU results in new information being loaded into the map.

Information relating to the actual number of map entries that have been loaded is retained in the CPU in order to prevent access to an entry in the map above that number. If a logical address of the process causes the CPU to generate a map index that is greater than the number of loaded entries, the CPU asserts a map fault trap.

3.9.3 Map Entries

Each 16-bit entry in the map RAM contains a valid bit (HMAPVALID), four write protect bits (HPRIV00 thru 03), and an 11-bit block address. The 11-bit address defines a specific memory block of 2K words. Each of the four write protect bits provides 512-word write protection granularity, i.e., any specified quarter of the addressed 2K-word block can be protected. This facility is used to prevent inadvertant destruction of information in the protected area designated.

3.9.4 Map Load

During firmware map load sequences (context switch time), data is routed from the cache or main memory one 32-bit word at a time. This data constitutes two 16-bit map entries, and it is selected via the cache multiplexer out onto the CAMUX bus. Data in the CAMUX00 thru 31 lines is clocked into the map in registers. These two registers are loaded in parallel by the HEVECAREGCK clock signal, and their outputs are enabled by the LLDMAPLEFT and LLDMAPRIGHT signals so that they may be individually written to the map RAM. The load and write enable signals are generated by MS Unit firmware.

If a map access is initiated, the relevant hit bit is not set, the information must be fetched from cache or main memory. A check is always made to see if the required map entries are cache resident. However, for a majority of map loads, the data will not be in the cache and a fetch from main memory is implemented. The map is then reaccessed for the required block address.

The time required to load the map is minimized by organization which uses a look-aside buffer technique. The hit map RAM functions as the look-aside buffer.

3.9.5 Look-aside Buffer

Load map time (context switch time) is minimized by using a look-aside buffer technique. The hit map RAM actually contains 1024 (1K) 16-bit locations; however, only 128 locations are used. Each single bit in the hit RAM corresponds to a register (16 bits) in the map RAM, and indicates whether the register has been loaded or not.

The look-aside buffer speeds up the time required to clear the map during a load sequence. Instead of sequentially clearing all 2048 locations in the map RAM, the locations in the hit RAM are cleared by loading a reset pattern set up on the YB bus. Each register in the hit RAM corresponds to 16 registers in the map RAM. Therefore, the map RAM can be completely cleared in 128 cycles. There is no need to actually clear the contents of the map RAM since the hit RAM bits provide the overriding factor and, in effect, signify whether the information in the map RAM is usable or not.

Bits in the hit RAM are checked by using the most significant 11 bits of the 24-bit logical address. The lower order seven bits (HMADDR08 thru 14) address one of the 128 locations, while bits HMADDR15 thru 18 are used as data select inputs at the 16:1 multiplexer to select one of the 16 bits from the addressed location. The selected bit is monitored by the map error logic and used to generate the HMAPMISS signal if appropriate (i.e., if the hit RAM bit signifies that the corresponding map RAM register has not been loaded). The HMAPMISS signal is divided into two separate signals which are routed to the microinterrupt logic on the IE unit. These two signals reflect the distinction between an operand map miss and an instruction map miss.

If an operand read or write results in a map miss, a data map miss (HDMAPMISS) signal is generated. This results in an immediate microinterrupt at the IE unit. A memory read sequence is initiated and the required pair of map entries are fetched from cache or main memory, validated, and loaded into the map. The macroinstruction pipeline is frozen until the map is reaccessed for the required data.

If an instruction fetch results in a map miss, a map miss error is associated with the instruction loaded in the I3 pipeline register. At the I2 decode stage of the pipeline, the IE unit initiates a memory read sequence. The required pair of map entries are fetched from cache or main memory. During this time, the macroinstruction pipeline is flushed. The map is then reaccessed for the required information, the pipeline is refilled, and the instruction stream is continued.

Data may be written to the hit RAM either over the YB bus or via bit generation logic. In both cases, the data is routed via the set map entry logic onto the LHITMAP00 thru 15 lines. The YB bus (HYB16 thru 31) is the data source when an initial configuration is loaded into the hit RAM or when the hit RAM is reset. The bit generation logic consists of a decoder which operates on logical memory address bits HMADDR15 thru 18 and decodes to one of 16 mutually exclusive outputs on the LSETMAP00 thru 15 lines. The hit RAM is updated by entering data into the appropriate register via parallel OR functions. This ensures that only the relevant bit in the 16-bit register is changed to signify that the corresponding map register has been loaded, the other 15 bits remain unchanged.

3.9.6 Map Bypass Multiplexing

A 2:1 multiplexer is used to select either the 11 most significant bits of the logical address (designated HMADDR08 thru 18) or the 11-bit output from the map RAM. The HMADDR08 thru 18 lines are selected only when the CPU is in the unmapped mode, or temporarily unmapped, due to IE unit firmware transfers or physical memory accesses. At all other times, the map RAM output is selected via the multiplexer.

3.9.7 Logical to Physical Address Conversion

Conversion from a logical to a physical address is performed by addressing the map RAM with the most significant 11 bits (HMADDR08 thru 18) of the 24-bit logical address. (The hit map RAM is addressed at the same time via the HMADDR08 thru 14 lines). The output produced on the map RAM I/O lines is an 11-bit physical address. After selection through the 2:1 map multiplexer, this 11-bit output is appended to the 13 least significant bits (designated HPADDR19 thru 31) to form the 24-bit physical memory address. This address is routed over the physical memory address bus and used to access the cache or main memory. The most significant 11 bits address a block of 2K locations, the next 11 bits address a specific location within the 2K block, and the two least significant bits may be used to specify one of the four bytes within the location.

CHAPTER 4

MICROPROGRAMMING

4.1 The Microword

The 64-bit, fifteen field microword provides the basic unit of control for the CONCEPT 32/67 CPU. Figure 4-1 illustrates the microword fields and their associated microinstructions. Subsequent tables of this chapter will give detailed descriptions of the individual microinstructions and their functions.

The Microsequencer (MS) Unit control store consists of thirty-two 2K by 4-bit programmable read-only-memories (PROMS), organized into a 64-bit wide by 4K-word double deep array. This control store is known as the control read-only-memory (CROM). A microword is burned into each CROM location (address), the sum of all microwords being the microprogram or firmware.

The CROM may be replaced by the alterable control store (ACS) or augmented by the writable control store (WCS) which are also located on the MS Unit board. The ACS and WCS are random-access-memory (RAM) rather than PROM and are under software control.

Although the microprogram counter (uPC), with its 16-bit wide path, is able to address any one of 64K different word locations, only the 12 least-significant uPC bits, 04 through 15, are needed to address the 4K-word PROM or ACS. Additional uPC bits 02 and 03 are used when addressing the WCS. The ACS is enabled through software control.

CONCEPT 32/67 logic drawings referred to in this chapter are the Microsequencer (MS) Unit, 130-103568, the Cache SelBUS (CS) Unit, 130-103583, and the Instruction Execution (IE) Unit, 130-103584.

4.2 Firmware Responsibility

The firmware is responsible for control of the CONCEPT 32/67 CPU hardware to accomplish the following operations:

1. Decode the macroinstructions into microinstruction sequences that perform the following:
    a. Fetch operands from memory and/or the register set.
    b. Modify the operands as specified by the macroinstruction.
    c. Set condition codes to indicate the results of the operand modification; that is, overflow, result zero, positive, or negative.
    d. Store the modified operands back into memory or the register set as required.
    e. Translate macroinput/output and interrupt control instructions into SelBUS physical addresses and transfer sequences (SelBUS protocol) that cause the macrolevel addressed channel/device to perform the required function; that is, input data, output data, enable interrupt, etc.
2. Monitor the macroexecution stream for the occurrence of errors (nonpresent memory, privilege violations, power fail, etc.). If an error does occur, the firmware must do the following:
    a. Identify the error.
    b. Prioritize the error.

c. Schedule the error by executing a trap sequence.

d. Perform a software context switch to a trap handler program.

3. Monitor the macroexecution stream for the occurrence of interrupts. If an interrupt occurs, the firmware must do the following:

    a. Identify the level of interrupt (priority).

    b. Execute an interrupt sequence.

    c. Perform a software context switch to a specific interrupt handler program.

4. Monitor the macroexecution stream for the occurrence of control panel communications. If control panel communication does occur, the firmware must do the following:

    a. Perform the requested function; that is, run, halt, etc.

    b. Translate the requested function into a SelBUS sequence (protocol) with the system control panel.

4.3 Machine Cycles

Each microinstruction uses two time periods for execution: the CROM cycle and the CREG (control register) cycle (see figure 4-2). The timing that regulates these cycles is a single-phase clock that triggers every 150 nanoseconds throughout the system.

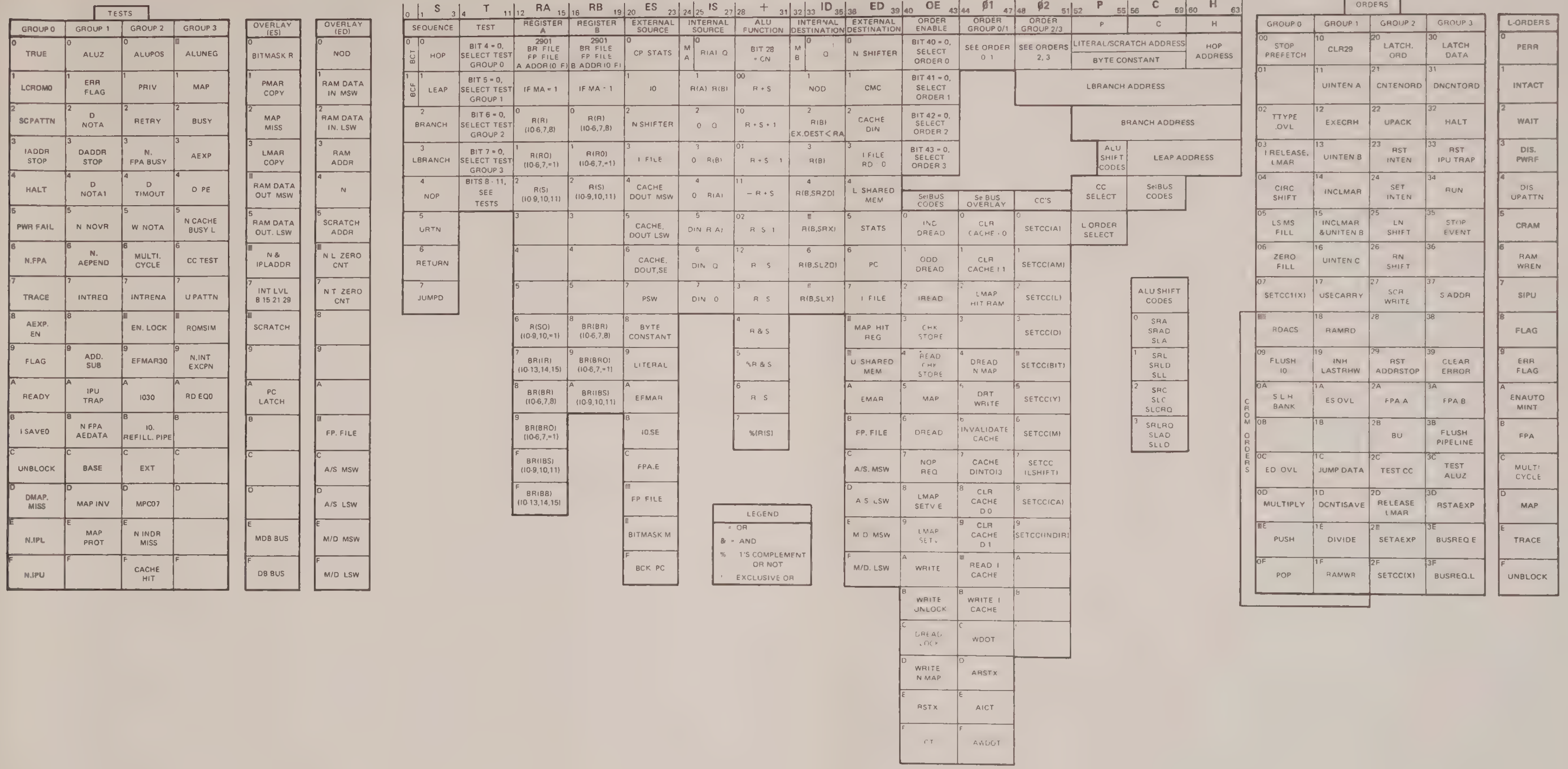
The CROM cycle refers to functions performed at the end of the control store access cycle. These functions are performed by direct output or decodes of the CROM without an intervening clock. Typically these include microsequencing and operation monitoring (test) functions.

The CREG cycle refers to functions performed from the control store register. CREG cycle functions require an intervening clock between them and CROM cycle functions and can provide a result-oriented operation for a full step (150 nsec) following the CROM step. Generally, CREG cycle functions control the data structure and data put-aways.

At the beginning of each CROM cycle the uPC, from its internal register, sends address bits which will fetch the next microword from the CROM. The contents of the microword (CROM bits) are fanned out to various locations in the CPU to arrange, in advance of the clock edge, functions that are to occur during the CREG cycle.

On the next clock, the CROM bits pass through the CREG flip-flops and fan out to control the entire data structure. All elements that operate synchronously with the ALU need the full 150 nanosecond cycle time, so everything in the data structure runs from the control register.

When an instruction N is fetched, it is clocked through a CROM cycle and a CREG cycle. When instruction N progresses from the CROM cycle to the CREG cycle, a subsequent instruction N+1 is clocked into the CROM cycle. Likewise, when instruction N is clocked into the CREG+1 cycle, instruction N+1 starts the CREG cycle, and a subsequently fetched instruction N+2 enters the CROM cycle. Testing of each instruction is done near the end of the CROM cycle, and the test results must be true about 10 nanoseconds before the end of the cycle. Tests related to the execution (CREG cycle) of instruction N must be coded in instruction N + 1. Execution related tests include ALUZ, ALUPOS, etc. CREG+1 cycle functions include cache and map transactions and setting of condition codes.



TESTS

Code	GROUP 0	GROUP 1	GROUP 2	GROUP 3
0	TRUE	ALUZ	ALUPOS	ALUNEG
1	LCROM0	ERR FLAG	PRIV	MAP
2	SCPATTN	D NOTA	RETRY	BUSY
3	IADDR STOP	DADDR STOP	N. FPA BUSY	AEXP
4	HALT	D NOTA1	D TIMOUT	D PE
5	PWR FAIL	N NOVR	W NOTA	N CACHE BUSY L
6	N.FPA	N. AEPEND	MULTI. CYCLE	CC TEST
7	TRACE	INTREQ	INTRENA	U PATTN
8	AEXP. EN		EN. LOCK	ROMSIM
9	FLAG	ADD. SUB	EFMAR30	N.INT EXCPN
A	READY	IPU TRAP	I030	RD EQ0
B	I SAVE0	N FPA AEDATA	IO. REFILL. PIPE	
C	UNBLOCK	BASE	EXT	
D	DMAP. MISS	MAP INV	MPC07	
E	N.IPL	MAP PROT	N INDR MISS	
F	N.IPU		CACHE HIT	

Code	OVERLAY (ES)
0	BITMASK R
1	PMAR COPY
2	MAP MISS
3	LMAR COPY
4	RAM DATA OUT MSW
5	RAM DATA OUT. LSW
6	N & IPLADDR
7	INT LVL 8 15 21 29
8	SCRATCH
9	
A	PC LATCH
B	
C	
D	
E	MDB BUS
F	DB BUS

Code	OVERLAY (ED)
0	NOD
1	RAM DATA IN MSW
2	RAM DATA IN. LSW
3	RAM ADDR
4	N
5	SCRATCH ADDR
6	N L ZERO CNT
7	N T ZERO CNT
8	
9	
A	
B	FP. FILE
C	A/S MSW
D	A/S LSW
E	M/D MSW
F	M/D LSW

Bits	0	1–3	4–11	12–15	16–19	20–23	24	25–27	28–31	32	33–35	36–39	40–43	44–47	48–51	52–55	56–59	60–63
Field		S	T	RA	RB	ES		IS	+		ID	ED	OE	Ø1	Ø2	P	C	H
Name		SEQUENCE	TEST	REGISTER A	REGISTER B	EXTERNAL SOURCE		INTERNAL SOURCE	ALU FUNCTION		INTERNAL DESTINATION	EXTERNAL DESTINATION	ORDER ENABLE	ORDER GROUP 0/1	ORDER GROUP 2/3	P	C	H

SEQUENCE (S)

Code	
0 (bit 0)	BCT
1 (bit 0)	BCF
0	HOP
1	LEAP
2	BRANCH
3	LBRANCH
4	NOP
5	URTN
6	RETURN
7	JUMPD

TEST (T)

- BIT 4 = 0, SELECT TEST GROUP 0
- BIT 5 = 0, SELECT TEST GROUP 1
- BIT 6 = 0, SELECT TEST GROUP 2
- BIT 7 = 0, SELECT TEST GROUP 3
- BITS 8 - 11, SEE TESTS

REGISTER A (RA)

Code	
	2901 BR FILE FP FILE A ADDR (0 F)
	IF MA = 1
0	R(R) (10-6,7,8)
1	R(RO) (10-6,7,=1)
2	R(S) (10-9,10,11)
3	
4	
5	
6	R(SO) (10-9,10,=1)
7	BR(IR) (10-13,14,15)
8	BR(BR) (10-6,7,8)
9	BR(BRO) (10-6,7,=1)
F	BR(IBS) (10-9,10,11)
F	BR(B8) (10-13,14,15)

REGISTER B (RB)

Code	
	2901 BR FILE FP FILE B ADDR (0 F)
	IF MA - 1
0	R(R) (10-6,7,8)
1	R(RO) (10-6,7,=1)
2	R(S) (10-9,10,11)
3	
4	
5	
8	BR(BR) (10-6,7,8)
9	BR(BRO) (10-6,7,=1)
A	BR(IBS) (10-9,10,11)

EXTERNAL SOURCE (ES)

Code	
0	CP STATS
1	IO
2	N SHIFTER
3	I FILE
4	CACHE DOUT MSW
5	CACHE. DOUT LSW
6	CACHE. DOUT,SE
7	PSW
8	BYTE CONSTANT
9	LITERAL
A	EFMAR
B	IO.SE
C	FPA.E
D	FP FILE
E	BITMASK M
F	BCK PC

INTERNAL SOURCE (IS) (bit 24: MA)

Code	
0	R(A) Q
1	R(A) R(B)
2	0 Q
3	0 R(B)
4	0 R(A)
5	DIN R(A)
6	DIN Q
7	DIN 0

ALU FUNCTION (+)

Code	
	BIT 28 = CN
00	R + S
10	R + S + 1
01	R + S 1
11	− R + S
02	R S 1
12	R S
3	R S
4	R & S
5	%R & S
6	R S
7	%(R S)

INTERNAL DESTINATION (ID) (bit 32: MB)

Code	
0	Q
1	NOD
2	R(B) EX.DEST < RA
3	R(B)
4	R(B,SRZD)
5	R(B,SRX)
6	R(B,SLZD)
7	R(B,SLX)

EXTERNAL DESTINATION (ED)

Code	
0	N SHIFTER
1	CMC
2	CACHE DIN
3	I FILE RD 0
4	L SHARED MEM
5	STATS
6	PC
7	I FILE
8	MAP HIT REG
9	U SHARED MEM
A	EMAR
B	FP. FILE
C	A/S. MSW
D	A S LSW
E	M D MSW
F	M/D. LSW

ORDER ENABLE (OE)

- BIT 40 = 0, SELECT ORDER 0
- BIT 41 = 0, SELECT ORDER 1
- BIT 42 = 0, SELECT ORDER 2
- BIT 43 = 0, SELECT ORDER 3

ORDER GROUP 0/1 (Ø1): SEE ORDER 0 1

ORDER GROUP 2/3 (Ø2): SEE ORDERS 2, 3

P, C: LITERAL/SCRATCH ADDRESS; BYTE CONSTANT

H: HOP ADDRESS

LBRANCH ADDRESS (bits 48–63)

BRANCH ADDRESS (bits 52–63)

ALU SHIFT CODES; LEAP ADDRESS

CC SELECT; SelBUS CODES

L ORDER SELECT

Code	SelBUS CODES	Se BUS OVERLAY	CC'S
0	INC DREAD	CLR CACHE I 0	SETCC(A)
1	ODD DREAD	CLR CACHE I 1	SETCC(AM)
2	IREAD	LMAP HIT RAM	SETCC(L)
3	CHK STORE		SETCC(D)
4	READ CHK STORE	DREAD N MAP	SETCC(BIT)
5	MAP	DRT WRITE	SETCC(Y)
6	DREAD	INVALIDATE CACHE	SETCC(M)
7	NOP REQ	CACHE DINTOI3	SETCC (LSHIFT)
8	LMAP SETV E	CLR CACHE D 0	SETCC(CA)
9	LMAP SET V	CLR CACHE D 1	SETCC(INDIR)
A	WRITE	READ I CACHE	
B	WRITE UNLOCK	WRITE I CACHE	
C	DREAD LOCK	WDOT	
D	WRITE N MAP	ARSTX	
E	RSTX	AICT	
F	CT	AWDOT	

Code	ALU SHIFT CODES
0	SRA SRAD SLA
1	SRL SRLD SLL
2	SRC SLC SLCRQ
3	SRLRQ SLAD SLLD

LEGEND	
=	OR
& =	AND
%	1'S COMPLEMENT OR NOT
'	EXCLUSIVE OR

ORDERS

GROUP 0	GROUP 1	GROUP 2	GROUP 3
00 STOP PREFETCH	10 CLR29	20 LATCH. ORD	30 LATCH DATA
01	11 UINTEN A	21 CNTENORD	31 DNCNTORD
02 TTYPE .OVL	12 EXECRH	22 UPACK	32 HALT
03 I RELEASE. LMAR	13 UINTEN B	23 RST INTEN	33 RST IPU TRAP
04 CIRC SHIFT	14 INCLMAR	24 SET INTEN	34 RUN
05 LS MS FILL	15 INCLMAR &UNITEN B	25 LN SHIFT	35 STOP EVENT
06 ZERO FILL	16 UINTEN C	26 RN SHIFT	36
07 SETCC1(X)	17 USECARRY	27 SCR WRITE	37 S ADDR
08 RDACS	18 RAMRD	28	38
09 FLUSH IO	19 INH LASTRHW	29 RST ADDRSTOP	39 CLEAR ERROR
0A S L H BANK	1A ES OVL	2A FPA A	3A FPA B
0B	1B	2B BU	3B FLUSH PIPELINE
0C ED OVL	1C JUMP DATA	2C TEST CC	3C TEST ALUZ
0D MULTIPLY	1D DCNTISAVE	2D RELEASE LMAR	3D RSTAEXP
0E PUSH	1E DIVIDE	2E SETAEXP	3E BUSREQ E
0F POP	1F RAMWR	2F SETCC(X)	3F BUSREQ.L

CROM ORDERS

Code	L-ORDERS
0	PERR
1	INTACT
2	WAIT
3	DIS. PWRF
4	DIS UPATTN
5	CRAM
6	RAM WREN
7	SIPU
8	FLAG
9	ERR FLAG
A	ENAUTO MINT
B	FPA
C	MULTI CYCLE
D	MAP
E	TRACE
F	UNBLOCK

830483

Figure 4–1. Microword

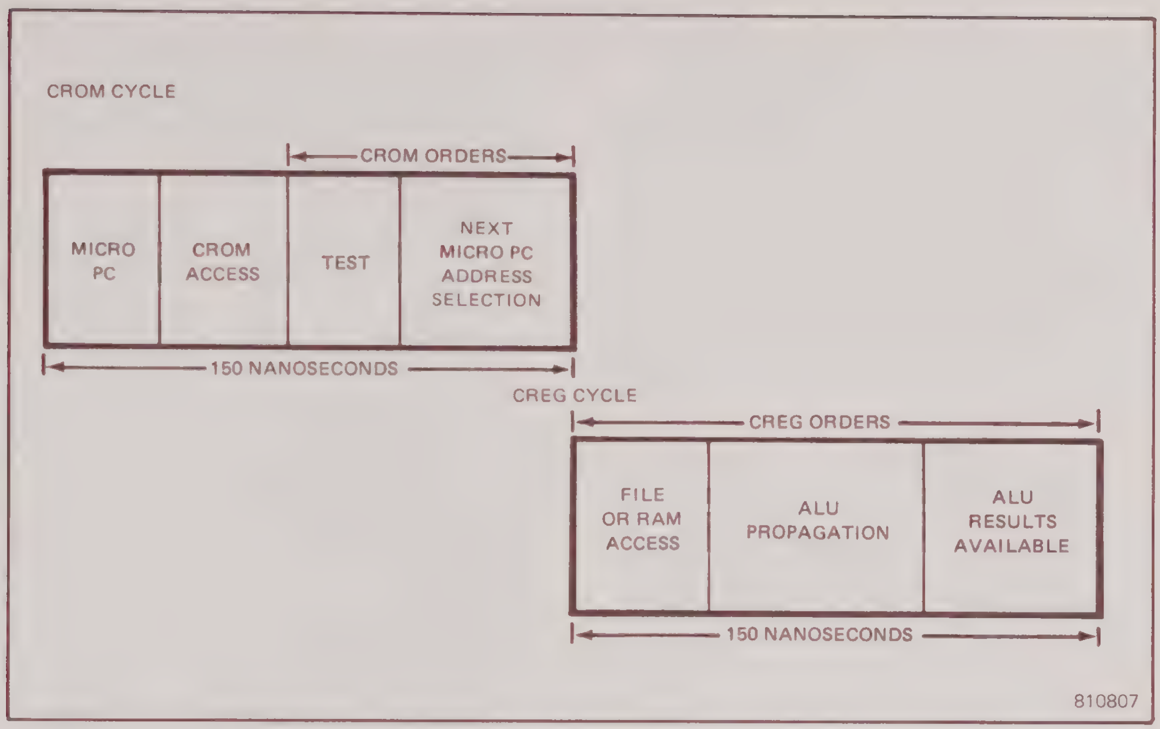

CROM CYCLE
CROM ORDERS
MICRO PC
CROM ACCESS
TEST
NEXT MICRO PC ADDRESS SELECTION
150 NANOSECONDS
CREG CYCLE
CREG ORDERS
FILE OR RAM ACCESS
ALU PROPAGATION
ALU RESULTS AVAILABLE
150 NANOSECONDS
810807

Figure 4-2. CROM Cycle and CREG Cycle Descriptions

4.4 Microword Field Descriptions

The following paragraphs describe the microword fields and their operation. For detailed information on the individual functions of these fields, see tables 4-1 through 4-25 at the end of this chapter.

4.4.1 Sequence Field

The sequence (S) field, CROM bits 00 through 03, specifies the details of the branch function (or sequence) when a branch is selected, and determines the test results required to enable that branch.

CROM bit 00, when low, generates a branch condition true and, when high, a branch condition false capability. That is, if branch condition false is selected and the test results are false, then the branch is taken. This is accomplished in the test structure where CROM bit 00 is used as the enable input of a 2:1 multiplexer and, inverted, as the enable input of a second 2:1 multiplexer. The 4-bit outputs of the two multiplexers are fed to the same inputs of an or gate and the resulting output is the HTESTTRUE signal to the uPC control. If the HTESTTRUE signal is true, the selected sequence is performed; if false, the next sequential step in the microprogram is fetched and executed.

CROM bit 01 is input, through a 2:1 multiplexer and an inverter, to the enable input of a decoder/demultiplexer. CROM bits 02 and 03 are used as the select bits to the decoder/demultiplexer and, with the CROM bit 01 low/true, select the inactive (NOP), the decode jump (JUMPD), the top of stack pickup (RETURN), or the return from microinterrupt (URTN).

There are four types of special branch instructions which select either 4, 8, 12, or 16 bits of the microword. To provide these, CROM bits 01 through 03 are also connected to a 4-bit shift register whose output, with CROM bit 01 low/false, is the select signals LHOLD01 through LHOLD03. These signals, through a series of gates, control which of CROM bits 48 through 63 are to be used to provide the address of the next microword when a branch is specified.

The sequence field can be considered part of the test structure since it controls some of the uPC operation. All sequence operations controlled by the sequence field are conditional, based on the test results and whether CROM bit 00 is in the branch condition true or branch condition false state.

4.4.2 Test Field

The test (T) field, CROM bits 04 through 11, determines which signals are to be tested, in the test structure, during the CROM cycle. The test field consists of four individual test group enable bits (CROM bits 04 through 07) and a 4-bit line number designator field (CROM bits 08 through 11).

The test structure is a matrix of input signals that are selected, by the test field bits, and multiplexed into a single signal (HTESTTRUE) which is sent to the uPC control. This signal determines whether a jump, if specified, will be taken. The test structure is capable of monitoring all of the internal conditions and states within the CPU and all of the external signals input to the CPU that may affect its operation.

The horizontal dimension of the test structure matrix is called a group and is controlled by CROM bits 04 through 07. The vertical dimension of the matrix is called a line number and is controlled by CROM bits 08 through 11.

Each of CROM bits 04 through 07, when low, can enable a specific group of tests by activating one of four 16:1 multiplexers in the test structure. CROM bits 08 through 11, the control inputs to the 16:1 multiplexers, determine the specific line of the chosen test group to be tested. Multiple groups may be enabled, but all tests selected must have the same line number.

For a multiple test BCT (CROM00=0), the test structure enables a branch if any one of the specified tests is high. If all specified tests are low, the test structure disables the branch.

Equation:
HTESTTRUE=TEST1:TEST2:TEST3:TE
ST4
Syntax:
IF TEST1:TEST2:TEST3:TEST4*GO
TO BRANCH

For a multiple test BCF (CROM00=1), the test structure enables a branch if any one of the specified tests is low. If all specified tests are high, the test structure disables the branch.

Equation:
HTESTTRUE=%TEST1:%TEST2:%TES
T3:%TEST4
Syntax:
IF%TEST1:TEST2:TEST3:TEST4*GO
TO BRANCH

4.4.3 Register A Field

The register A (RA) field, CROM bits 12 through 15, determines the A-port address of one of the 16 registers (0 through F) in the uP2901, the base register file and the floating-point accelerator (FPA) file. Additionally, the value of CROM bit 24, the macro A bit, controls the source of the A-port address.

When the MA bit is zero, the hexadecimal value of CROM bits 12 through 15 directly represents an A-port address in the range of 0 through F.

When the MA bit is one, CROM bits 12 through 15 are used with a multiplexer which selects bit combinations from the IMUX Bus and uses their hexadecimal value to determine the A-port address. The IMUX Bus can source either the I1 or I0 register, depending on the state of the HADVANCEI0 signal.

The A-port addresses are determined at the end of the CROM cycle and used during the CREG cycle. Since data can only be read from the A-port, the RA field register can only be used as a source.

4.4.4 Register B Field

The register B (RB) field, CROM bits 16 through 19, determines the B-port address of one of the 16 registers (0 through F) in the uP2901, the base register file and the floating- point accelerator (FPA) file. Additionally, the value of CROM bit 32, the macro B bit, controls the source of the B-port address.

When the MB bit is zero, the hexadecimal value of CROM bits 16 through 19 directly represents a B-port address in the range of 0 through F.

When the MB bit is one, CROM bits 16 through 19 are used with a multiplexer which selects bit combinations from the IMUX Bus and uses their hexadecimal value to determine the B-port address. The IMUX Bus can source either the I1 or I0 register, depending on the state of the HADVANCEI0 signal.

The B-port addresses are determined at the end of the CROM cycle and used during the CREG cycle. Since data can be both written into and read from the B-port, the RB field register can be used a both a source and a destination.

4.4.5 External Source Field

The external source (ES) field, CROM bits 20 through 23, selects the DB or MDB Bus data source to be fed to the direct data inputs, D0 through D3, of the uP2901s, and to the scratchpad.

CROM bits 20 through 23 are input to the DB and MDB Bus control circuitry on the MS, IE, and CS Units. From the control circuitry, the enable signals to the selected data source are generated and held, in latching flip-flops, until the next clock (CREG cycle). This latching method assures proper timing of these signals so that all coordinating sources are enabled within a nominal margin.

The ES field can address 16 possible external sources; however, an additional 16 sources can be chosen through use of the ES overlay (ES.OVL) CROM order.

4.4.6 Internal Source Field

The internal source (IS) field, CROM bits 24 through 27, determines the sources, within the uP2901, of the two 4-bit ALU input words R and S.

CROM bit 24, though part of the IS field, is used as the macro A (MA) bit by the RA field (paragraph 4.4.3). The MA bit controls the source of the A-port address.

CROM bits 25 and 27 are fed to flip-flops where they are converted to CREG cycle signals. If a multiply or divide order has not been given, CROM bit 26 passes through the ALU shift multiplexer and a flip-flop, where it also becomes a CREG cycle signal. These CREG cycle signals are then fed, in parallel, to the I0 through I2 inputs of the eight uP2901s where they control the ALU source operand decode.

The ALU performs its logic and arithmetic operations on two 4-bit input words R and S. The R-input field is driven from a 2-input multiplexer, while the S field is driven from a 3-input multiplexer. Both multiplexers also have an inhibit capability; that is, no data is passed. This is equivalent to a zero source operand.

The ALU R-input multiplexer has the RAM A-port and the direct data inputs (D) connected as inputs. Likewise, the ALU S-input multiplexer has the RAM A-port, the RAM B-port and the Q register connected as inputs. This multiplexer scheme gives the capability of selecting various pairs of the A, B, D, Q and 0 inputs as source operands.

4.4.7 ALU Function Field

The ALU function (+) field, CROM bits 28 through 31, selects one of eight ALU functions to be performed by the uP2901. These functions include three binary arithmetic and five logic operations.

CROM bit 28, the most-significant bit of the field, is the Cn bit. This bit serves as the carry-in bit to the least-significant uP2901. The carry-in bit is used for arithmetic operations only, having no effect on logic operations. If an ALU function requires, or uses, the ones complement, the carry-in will not be used; however, if a function requires the twos complement, the carry-in will be given.

CROM bits 29 through 31 are fed to inputs of the CREG flip-flops. On the next CREG clock, these CREG cycle signals feed the I3 through I5 inputs of all eight uP2901s where their octal value is decoded and used to select the internal ALU function.

4.4.8 Internal Destination Field

The internal destination (ID) field, CROM bits 32 through 35, determines the destination, within the uP2901, of the results of the ALU operation. The ALU results are also sent to the uP2901's Y-port output.

CROM bit 32, though part of the ID field, is used as the macro B (MB) bit by the RB field (paragraph 4.4.4). The MB bit controls the source of the B-port address.

CROM bits 33 through 35 are clocked through CREG flip-flops where they become CREG signals HIC06 through HIC08. These signals feed the I6 through I8 inputs of the eight uP2901s where their octal value is decoded and used to select the ALU destination.

In adddition, signal HIC07 is used to control the ALU shift multiplexers when an ALU shift is specified, and signal HIC08 is used by the condition control logic in indicating a shifter overflow.

4.4.9 External Destination Field

The external destination (ED) field, CROM bits 36 through 39, selects the Y or YB Bus data destination to be loaded from the Y-port outputs, Y0 through Y3, of the uP2901s.

CROM bits 36 through 39 are clocked through CREG flip-flops into the CREG cycle. The resulting CREG bits are then input into the Y Bus control circuitry on the MS, IE and CS units. From the control circuitry, a load signal is sent to the specified destination and the Y or YB Bus bits are loaded.

The ED field can address 16 possible external destinations; however, an additional 16 destinations can be chosen through use of the ED overlay (ED.OVL) CROM order.

4.4.10 Order Enable and Order Group Fields

The order enable (OE) and order group (O1 and O2) fields, CROM bits 40 through 51, determine which orders are to be enabled in the order structure.

The order structure is an output signal array that decodes microwords into individual output signals. From one to four orders may be generated from a single microword. In general, orders are used to initialize or terminate internal control functions. Orders may also serve as qualifiers for certain events and register strobe signals.

The order structure provides both CROM and CREG cycle orders. Some orders are delayed for an additional clock to provide CREG+1 cycle orders.

The horizontal dimension of the order array is called a group and is controlled by CROM bits 40 through 43. The vertical dimension of the array is called a line number and is controlled by CROM bits 44 through 51.

Each of CROM bits 40 through 43, when low, can select a specific group of orders by enabling one of twelve decoder/demultiplexers in the IE and MS units. Up to four different groups may be enabled at one time.

The O1 and O2 fields, CROM bits 44 through 47 and 48 through 51 respectively, select the individual line numbers within the enabled groups. These CROM bits are input to the decoder/demultiplexers and their hexadecimal value determines the chosen line numbers. Since group 0 and 1 line numbers are controlled by the O1 field and group 2 and 3 line numbers are controlled by the O2 field, it is possible to select two different line numbers with one microword.

4.4.11 P, C and H Fields

The P, C and H fields, CROM bits 52 through 63, are able to perform multiple functions, including addressing and code selection. These fields are used singly, or in combination with each other, to provide the various field-lengths required. This combining of fields allows the microword to perform several functions simultaneously if their binary codes agree. The following paragraphs describe these field combinations and their functions.

4.4.11.1 L Branch Address Field

The long branch address field, CROM bits 48 through 51, combines the O2, P, C and H fields. The contents of this field are used to replace the entire sixteen bits of the uPC. The long branch function is provided for the possibility of add-on RAM (WCS). The limit of the long branch is any of 64K lines in the @0000 through FFFF range.

4.4.11.2 Branch Address Field

The branch address field, CROM bits 52 through 63, combines the P, C and H fields. The contents of this field are used to replace the twelve least-significant bits of the uPC during the CROM cycle, leaving the remaining four bits unchanged. Since the limit of this function is any of 4096 lines in the @000 through FFF range, a BRANCH will allow a sequence from any point in the firmware to any other point.

4.4.11.3 Leap Address Field

The leap address field, CROM bits 56 through 63, combines the C and H fields. The contents of this field are used to replace the eight least-significant bits of the uPC during the CROM cycle, leaving the remaining eight bits unchanged. This allows a branch to any of 256 lines in the @00 through FF range.

4.4.11.4 Hop Address Field

The hop address field, CROM bits 60 through 63, uses only the H field. The contents of this field are used to replace the four least-significant bits of the uPC during the CROM cycle, leaving the remaining twelve bits unchanged. This allows a branch to any of sixteen lines in the @0 through F range.

4.4.11.5 Literal/Scratch Address Field

The literal/scratch pad address field, CROM bits 52 through 59, combines the P and C fields. This field provides an address for either the scratch pad or the 32-bit constant (literal) PROM.

4.4.11.6 Byte Constant Field

The byte constant field, CROM bits 52 through 59, combines the P and C fields. This field provides a byte literal for the byte constant register in the IE unit.

4.4.11.7 CC Select Field

The condition code select field, CROM bits 52 through 55, uses only the P field. The hexadecimal value of this field determines the rules to be used in setting the condition codes. This field is enabled by the SETCC(X) (group 2, line F) or SETCC1(X) (group 0, line 7) CREG orders.

4.4.11.8 L-Order Select Field

The level order select field, CROM bits 52 through 55, uses only the P field. Level orders differ from other firmware orders in that, once set, they remain in effect until reset. The hexadecimal value of this field determines the level order to be set or reset. This field is enabled, and the level order set, by the order LATCH.ORDER (group 2, line 0). Level orders are reset by giving the orders LATCH.ORDER and LATCH.DATA (group 3, line 0) simultaneously.

4.4.11.9 ALU Shift Codes Field

The ALU shift codes field, CROM bits 54 and 55, uses the two least-significant bits of the P field. In ALU shift operations, the value of this field determines the type and direction of shift and the ALU registers involved. The field, enabled when CROM bit 33 is equal to one, is used in conjunction with the internal destination field (CROM bits 32 through 35).

4.4.11.10 SelBUS Codes Field

The SelBUS codes field, CROM bits 56 through 59, uses only the C field. The hexadecimal value of this field determines the type of transfer to take place on the SelBUS. The field is enabled by either the BUSREQ.E order (group 3, line E), which uses the execution memory address register (EMAR) for the transfer, or the BUSREQ.L order (group 3, line F), which uses the logical memory address register (LMAR).

Any one of sixteen possible SelBUS codes may be addressed by this field; if the TTYPE.OVL order (group 0, line 2) is given at the same time, a different set of SelBUS codes (SelBUS overlay codes) may be addressed.

Table 4-1
Sequence Field CROM 00 through 03 (Sheet 1 of 2)

Reference:	MS Unit logic drawing 130-103568, sheets 23 through 25
Influenced by:	Results of the basic test field
Influences:	Address interpretation of CROM bits 48 through 63 and the next uPC address
General:	The binary value of CROM bit 00 determines the test results required to enable the selected branch or sequence. If CROM bit 00 = 0, Branch condition true If CROM bit 00 = 1, Branch condition false The octal value of CROM bits 01 through 03 determines the details of the branch function (or sequence) when a branch is selected.

Value	Syntax	Detailed Definition
0	*HOP	Hop. The HOP command causes the four bits of the H field (CROM bits 60-63) to replace the four least-significant bits of the uPC during the CROM cycle. The other twelve uPC bits remain unchanged. Limit of HOP: Any of 16 lines in the 0-F range.
1	*LEAP	Leap. The LEAP command causes the eight bits of the C and H fields (CROM bits 56-63) to replace the eight least-significant bits of the uPC during the CROM cycle. The other eight uPC bits remain unchanged. Limit of LEAP: Any of 256 lines in the 00-FF range.

Table 4-1
Sequence Field CROM 00 through 03 (Sheet 2 of 2)

Value	Syntax	Detailed Definition
2	*BRANCH	Branch. The BRANCH command causes the twelve bits of the P, C and H fields (CROM bits 52-63) to replace the twelve least-significant bits of the uPC during the CROM cycle. The other four uPC bits remain unchanged. Limit of BRANCH: Any of 4096 lines in the 000-FFF range. Locations 000-FFF represent 4K of CROM; therefore, in the CONCEPT 32/67 CPU, the BRANCH will allow a sequence from any point in the firmware to any other point.
3	*LBRANCH	Long branch. The LBRANCH command causes the sixteen bits of the O2, P, C and H fields (CROM bits 48-63) to replace all sixteen bits of the uPC. Limit of LBRANCH: Any of 64K lines in the 0000-FFFF range. The LBRANCH is provided for the addressing of WCS. The LBRANCH is also used during the JUMP.DATA CROM order (group 1, line C) which selects the 16 least-significant YB Bus bits (bits 16 through 31) as the branch address if the test is true.
4	*NOP	No operation. The NOP command causes no sequence operation to be performed. The uPC is incremented by one.
5	*URTN	Microreturn. The URTN command is used after a micro-interrupt has been serviced. The URTN causes the contents of the uPC save register to be the address of the next microinstruction.
6	*RETURN	Return. The RETURN command operates in conjunction with a link and push. The link causes the desired branch, and the push puts the present uPC+1 address into the FILO-type push/pop address stack within the uPC. The address stack is four deep. The RETURN command causes the last address placed in the stack to be used as the next uPC address after a return from a subroutine. Thus, the execution will continue on the next microinstruction after the last link and push. A pop order must also be issued with the return to eject the last address from the stack and make it available to the uPC's internal register.
7	*JUMPD	Decode jump. The JUMPD command causes the pipeline to be advanced (bumped). This occurs regardless of the test coded with the JUMPD. The specified test condition (true or false) only controls the uPC input multiplexer and determines if the decode vector (test true) or the uPC+1 (test false) is taken. If the test is false, the decode vector for the instruction moving into the I1 register, at the end of the JUMPD CROM cycle, is lost.

Table 4-2
Test Field CROM 04 through 11

Reference:	IE Unit logic drawing 130-103584, sheet 31
Influenced by:	Bit 00 of the sequence field
Influences:	Bits 01 through 03 of the sequence field
General:	The binary value of CROM bits 04 through 07 determines the test matrix groups to be enabled. CROM bits 04 through 07 are low/true; therefore @F indicates that no test is enabled. Any number of test groups may be enabled simultaneously. CROM bit 04 = 0 Test group 0 enabled CROM bit 05 = 0 Test group 1 enabled CROM bit 06 = 0 Test group 2 enabled CROM bit 07 = 0 Test group 3 enabled The hexadecimal value of CROM bits 08 through 11 determines the appropriate line of the test matrix. See paragraphs 4.4.2 for examples of multiple test equations and syntaxes. See tables 4-3 through 4-6, test groups 0 through 3 respectively, for detailed definitions of the individual tests.

Table 4-3
Test Group 0 (Sheet 1 of 3)

Value	Syntax	Signal	Detailed Definition
0	TRUE	+V(4)	The TRUE test is used to force a true test condition. This test is used for unconditional branches.
1	LCROM0	LCROM00	CROM bit 00 is low/true. This test indicates that the output of the CROM bit 00 NAND gate is low/true; that is, branch condition false has been selected. See the TEST.CC (group 2, line C) and TEST.ALUZ (group 3, line C) orders.
2	SCPATTN	HSCPATTNL	System control panel attention, latched. This test indicates that the IOP panel function needs servicing by the processor. The indication is generated by the IOP console when the SelBUS system control panel signal is on for more than one cycle. The HSCPATTNL signal is cleared by the CPU with the UPACK order (group 2, line 2).
3	IADDRSTOP	HIOIADDR-STOP	Instruction address stop. The instruction in the I0 register has an address stop pending.
4	HALT	HCPUHALT	CPU halt. This test indicates that the CPU is halted and microinstructions are no longer being executed. The test is driven from the reset output of the run/halt flip-flop. The flip-flop is set by the RUN order (group 3, line 4) or the IOP panel, and reset by the HALT order (group 3, line 2) or the IOP panel.
5	PWRFAIL	HPWRFAIL	Powerfail. This test indicates that a powerup or powerdown sequence has occurred. When powerfail is tested at CROM location 000, it is a powerup sequence; in all other cases, it is a powerdown. Powerfail can be inhibited by level order 3 (DIS.PWRF).

Table 4-3
Test Group 0 (Sheet 2 of 3)

Value	Syntax	Signal	Detailed Definition
6	N.FPA	LFPAPRESENT	Floating-point accelerator is not present. This test indicates that the floating-point accelerator is not on-line. The test is driven from the enable/disable switch, S1, on the floating-point accelerator, or by setting or resetting the FPA level order (value B).
7	TRACE	HTRACEL	Trace, latched. This is a test of the set output of level order E (TRACE). A positive test result indicates that the CPU is in an instruction-step (single-step) mode.
8	AEXP.EN	HAEXCPEN	Arithmetic exception trap enabled. This test indicates that the CPU is operating and should detect an arithmetic exception if one exists. The arithmetic exception trap is enabled by bit 07 of PSD1.
9	FLAG	HFLAGL	Flag, latched. This is a test of the set output of level order 8 (FLAG). The FLAG indicates that the context of the CPU has been temporarily altered, due to an interrupt, and must be restored before processing is resumed.
A	READY	HRDY	I/O response ready. This is a test of the set output of the ready flip-flop. The test indicates that the I/O channel, addressed by a previous advance transfer SelBUS code (AICT or ARSTX), has responded with a ready signal. The ready flip-flop is reset by either AICT or ARSTX, set by the LREADYX signal from the SelBUS, and preset by the CLEAR.ERROR order(group 3, line 9).
B	ISAVE0	HISAVE0	I0 register bits 11 through 15 are equal to zero. This test indicates when bits 11 through 15 of register I0 are equal to zero. The test is used with shift instructions to identify shift count zero. The register is decremented by use of the DNCNTISAVE order (group 1, line D).

Table 4-3
Test Group 0 (Sheet 3 of 3)

Value	Syntax	Signal	Detailed Definition
C	UNBLOCK	HUNBLOCK	Interrupts unblock. This is a test of the set output of level order F (UNBLOCK). The test indicates when interrupts are unblocked. The test is used for PSD2 bits 16 and 17. The unblock interrupt latch is controlled by software, using the BEI (block external interrupts) and UEI (unblock external interrupts) macroinstructions.
D	DMAP.MISS	HDMAPMISS	Data map miss. This test indicates that the CPU's map needs to be updated. The test is true after an access (read or write) has been attempted to an address that has not been loaded into the CPU's map. The test is driven from the cache transaction status logic. The signal is not true until one cycle after the cache transaction.
E	N.IPL	LIPLSWL	Not initial program load, latched. This test indicates that a firmware initial program load (IPL) is to be performed. The test is driven from the turnkey panel interface circuit in the MS Unit.
F	N.IPU	LIPUMODEL	Not IPU mode, latched. This test indicates that the IPU mode has not been selected. The test is driven from the PROCESSOR SELECT switch logic in the MS unit.

Table 4-4
Test Group 1 (Sheet 1 of 3)

Value	Syntax	Signal	Detailed Definition
0	ALUZ	HFE0L	ALU output equal to zero, latched. This test indicates that the 32-bit output of the uP2901s is equal to zero. This test is valid one clock after the ALU function and will remain valid for one clock. The ALUZ test should be coded two clocks after the ALU operation is coded.
1	ERR.FLAG	HERRFLAGL	Error flag, latched. This is a test of the set output of level order 9 (ERR.FLAG). The firmware uses this flag when exiting sub-routines to indicate that a memory or I/O error was detected in the sub-routine.
2	D.NOTA	HDNOTA	Data fetch, no transfer acknowledge. This test indicates that the memory operand or I/O channel addressed by the last memory operand read or I/O bus transfer was not present (non-present memory or I/O channel). This condition is cleared by the CLEAR.ERROR order (group 3, line 9). The test is driven from the cache data out status register in the CS Unit. This test is equivalent to the D.NOTA1 test (group 1, line 4).
3	DADDRSTOP	HDADDR-STOP	Panel data address stop. This test indicates that the panel has detected an operand (data) write or read address stop. The stop normally relates to the previous instruction. The address stop is cleared by the CLEAR.ERROR order (group 3, line 9).
4	D.NOTA1	HDNOTA	Data fetch, no transfer acknowledge. This test indicates that the memory operand or I/O channel addressed by the last memory operand read or I/O bus transfer was not present (non-present memory or I/O channel). This condition is cleared by the CLEAR.ERROR order (group 3, line 9). The test is driven from the cache data out status register in the CS Unit. This test is equivalent to the D.NOTA test (group 1, line 2).

Table 4-4
Test Group 1 (Sheet 2 of 3)

Value	Syntax	Signal	Detailed Definition
5	N.NOVR	NOVRFLOW	Not N-counter overflow. This test indicates that the contents of the N-counter are not equal to @FF in the up-count mode, and not equal to @00 in the down-count mode. See the CNTENORD (group 2, line 1) and DNCNTORD (group 3, line 1) orders).
6	N.AEPEND	LFPAAEPEND	Not arithmetic exception pending. This test indicates that an arithmetic exception has not occurred in the hardware floating-point accelerator during a floating-point instruction.
7	INTREQ	HINTREQ	Interrupt request. This test indicates that an interrupt is requesting service. This signal may be inhibited by the reset conditions of latch order F (UNBLOCK) and the interrupt enable flip-flop.
8			Not Used.
9	ADD.SUB	LFPAMDATA	Add/subtract operation. This test indicates that either an add or a subtract operation is taking place in the floating-point accelerator.
A	IPUTRAP	HIPUTRAP	Internal processing unit trap. This is a test of the set output of the IPU trap flip-flop in the MS Unit. The test indicates that the flip-flop has been set by a SIPU signal (level order 7) from another processor. The flip-flop is reset by the RSTIPUTRAP order (group 3, line 3).
B	N.FPA. AEDATA	LFPAAEDATA	Not floating-point arithmetic exception. This test indicates that the currently addressed floating-point accelerator registers do not contain arithmetic exception data.
C	BASE	HBASEMODE	Base register mode. This test indicates that the CPU is operating in the base register mode. The base register mode is enabled by bit 06 of PSD1.

Table 4-4
Test Group 1 (Sheet 3 of 3)

Value	Syntax	Signal	Detailed Definition
D	MAP.INV	HDMAPINV	Map data invalid. This test indicates that the current map data entry is invalid. The test is driven from the cache data out status register in the CS Unit.
E	MAPPROT	HDPROERR	Map data entry is protected. This test indicates that a write transaction has been attempted to a memory address that is write protected. The test is driven from the cache data out status register in the CS Unit. This test is valid only when the CPU is in the unprivileged mode.
F			Not Used.

Table 4-5
Test Group 2 (Sheet 1 of 3)

Value	Syntax	Signal	Detailed Definition
0	ALUPOS	LSIGNL	ALU output is positive, latched. This test indicates that the ALU sign bit (ALU bit 0) is a logical zero. The test is valid one clock after the ALU function and will remain valid for one clock. The ALUPOS should be coded two clocks after the ALU operation is coded.
1	PRIV	HPRIV	Privileged mode. This test indicates that the CPU is operating in the privileged mode. This mode is enabled by bit 00 of PSD1.
2	RETRY	HRETRY	Retry SelBUS response. This is a test of the set output of the retry flip-flop. The test indicates that an attempted SelBUS transaction has failed to occur and the CPU must try the transaction again. This condition is reset by the CLEAR.ERROR order (group 3, line 9).
3	N.FPABUSY	LFPABUSY	Not floating-point accelerator busy. This test indicates that the floating-point accelerator is not busy.
4	D.TIMOUT	HDTIMOUT	Data time out. This is a test of the cache data out status register. The test indicates that, during a SelBUS read sequence, the data return transfer (DRT) counter has timed out before receiving a DRT signal from the addressed device. This condition is cleared by the CLEAR.ERROR order (group 3, line 9).
5	W.NOTA	HWRNOTA	Data write, no transfer acknowledge. This test is driven by the write error status flip-flop. The test indicates that the transfer acknowledge (TA) signal was not received from the addressed device during a data write operation. This condition is cleared by the CLEAR.ERROR order (group 3, line 9).

Table 4-5
Test Group 2 (Sheet 2 of 3)

Value	Syntax	Signal	Detailed Definition
6	MULTI. CYCLE	HMULTICYCLE	Multiple-cycle instruction. This is a test of the set output of level order C (MULTI.CYCLE). The test indicates that the firmware is in the process of executing a multiple-cycle memory reference type instruction (e.g., LF, STF, CALL, RETURN).
7	INTRENA	HINTREN	Interrupt enable. This is a test of the set output of the interrupt enable flip-flop. This test indicates that interrupts are enabled. The set output of the interrupt enable flip-flop also influences the results of the INTREQ, IPUTRAP and UPATTN tests. The flip-flop is set by the SETINTEN order and reset by the RSTINTEN order.
8	EN.LOCK	HENLOCK	Enable lock transfer. This test is driven from the lower shared memory limit register. The register is enabled when the L.SHARED.MEM (value 04) external destination is chosen. This test indicates to the firmware that the software has executed a shared memory control macroinstruction, and the CPU is operating in a read and lock environment. This will cause the firmware to issue a DREAD.LOCK (SelBUS code C) transfer instead of a DREAD (SelBUS code 6) transfer.
9	EFMAR30	HI0EFFLMAR30	Effective MAR bit 30. This test indicates that effective memory address register bit 30 is set. The test is driven from the I0 register.
A	I030	HI0B30	I0 register bit 30. This test indicates that I0 register bit 30 is set. The test is driven from the I0 register.
B	I0REFILL. PIPE	HI0REFILLPIPE	I0 register, refill pipeline. This test indicates that the pipeline needs to be refilled. The test also indicates a pipeline violation where a memory store target has been prefetched into the pipeline. The test is driven from the I0 register.

Table 4-5
Test Group 2 (Sheet 3 of 3)

Value	Syntax	Signal	Detailed Definition
C	EXT	HEXT	Extended addressing mode. This test indicates that the CPU is operating in the extended addressing mode. The extended addressing mode is enabled by bit 05 of PSD1.
D	MPC07	HMPC07	Macroprogram counter bit 07. This test indicates that macroprogram counter bit 07 is set, indicating a macroprogram counter overflow. The test is driven from the macroprogram counter.
E	N.INDR. MISS	LI0NIDRMISS	Not indirect map miss. This test indicates that a map miss has occurred as a result of an indirect memory reference type instruction. The test is driven by the I0 status register.
F	CACHE.HIT	LMISST	Cache hit. This test, used for micro-diagnostic purposes, indicates that a memory read has resulted in a cache hit. The test is driven from the cache control circuitry.

Table 4-6
Test Group 3 (Sheet 1 of 2)

Value	Syntax	Signal	Detailed Definition
0	ALUNEG	HSIGNL	ALU output is negative, latched. This test indicates that the ALU sign bit (ALU bit 0) is a logical one. The test is valid one clock after the ALU function and will remain valid for one clock. The ALUNEG should be coded two clocks after the ALU operation is coded.
1	MAP	HMAPL	Mapped mode, latched. This is a test of the set output of level order D (MAP). The test is used to determine when the CPU is operating in the mapped mode. The mapped mode is enabled by bit 00 of PSD2.
2	BUSY	HBUSY	Device is busy. This is a test of the set output of the busy flip-flop. The test indicates that a SelBUS transaction failed to take place because the addressed device was busy. This condition is cleared by the CLEAR.ERROR order (group 3, line 9).
3	AEXP	HAEXP	Arithmetic exception. This is a test of the set output of the arithmetic exception flip-flop. The test indicates that an arithmetic exception has occurred. The flip-flop is set by either the output of the condition code field-programmable logic array (FPLA) or the SETAEXP order (group 2, line E). The flip-flop is reset by the RSTAEXP order (group 3, line D). This test excludes FPA arithmetic exceptions.
4	D.PE	HDPE	Data parity error. This test indicates that a parity error has been detected during a data read transaction from memory. This condition is cleared by the CLEAR.ERROR order (group 3, line 9). The test is driven from the cache data out status register.
5	N.CACHE. BUSY	LCACHE-BUSYL	Cache not busy, latched. This test indicates that the cache memory is not busy. The test is driven from the I/O clock circuitry in the CS Unit, and latched in the IE Unit.

Table 4-6
Test Group 3 (Sheet 2 of 2)

Value	Syntax	Signal	Detailed Definition
6	CCTEST	HCCTEST	Condition code test. This test is driven from the condition code logic in the IE Unit. A true test result indicates that one or more of the condition codes, in the tested combinations listed below, are equal to one. The octal value of I0 register bits 06 through 08 determines the CC combinations to be tested. I0 bits 06-08 — Combination tested 0 — I0 register bit 04 1 — CC1 2 — CC2 3 — CC3 4 — CC4 5 — CC2:CC4 6 — CC3:CC4 7 — CC1:CC2:CC3:CC4
7	UPATTN	HATTN	User panel attention (console interrupt). This test indicates tht the IOP attention function on the IOP console CRT has been executed, and the panel is calling for attention. The test is driven from a flip-flop in the turnkey panel interface circuit.
8	ROMSIM	HROMSIM	ROM simulator mode. This test indicates that the MS unit is operating with a Development Support System (DSS) or Instrumentation Interface Unit (IIU) attached, and ACS is under DSS or IIU control, replacing PROMs.
9	N.T. EXCPN	LINTEXCPN	Not interrupt exception. This test, when low, true, indicates that the MS unit detects a power fail, system panel attention, CPU halt, interrupt request, IPU trap, or console attention exception condition.
A	RD.EQ0	HI0RDEQU0	RD field is equal to zero. This test indicates that the I0 RD field (bits 06 through 08) is equal to zero.
B-F			Not used.

Table 4-7
Register A Field CROM 12 through 15 (Sheet 1 of 2)

Reference: IE Unit logic drawing 130-103584, sheet 25

General: If MA (CROM bit 24) = 0, the hexadecimal value of CROM bits 12 through 15 directly represents the A-port address of one of the 16 registers (0 through F) in the uP2901s, the IE Unit base register file, and the floating-point accelerator (FPA) file.

If MA = 1, CROM bits 12 through 15 are used with a multiplexer which selects bits from the IMUX Bus and uses them to determine the A-port address of one of the 16 registers (0 through F) in the uP2901s, the IE Unit base register file, and the FPA file. Inputs to the multiplexer are detailed in the following table.

File addresses are picked up at the end of the CROM cycle and used in the CREG cycle. If the instruction in the I1 register is moving into the I0 register (HADVANCE I0) during the CROM cycle, then the I1 register is the file address source; otherwise, the I0 register is the source.

To be construed as an RA function by the assembler, the syntax must appear to the right of the =sign. The IR ( ) or BR ( ) syntax also forces a value of 3 in the ES field (CROM bits 20 through 23); the FPR ( ) syntax forces a value of D into the ES field.

In the uP2901, the LSB is 00 and the MSB is 03.

Value	Syntax	Detailed Definition
0	R(R) or IR(R) or BR(R) or FPR(R)	A-port address bits 00 through 03 are driven by IMUX Bus bits 08, 07, 06 and CROM bit 12 respectively. CROM bit 12 is equal to zero. Used with arithmetic or logical add, subtract, multiply, divide, load and store instructions.
1	R(RO) or IR(RO) or BR(RO) or FPR(RO)	A-port address bits 01 through 03 are driven by IMUX Bus bits 07, 06 and CROM bit 12 respectively. Address bit 00 is driven to a logical one. CROM bit 12 is equal to zero. Used to select an odd register during doubleword instructions.
2	R(S) or IR(S) or BR(S) or FPR(S)	A-port address bits 00 through 03 are driven IMUX Bus bits 11, 10, 09, and CROM bit 12 respectively. CROM bit 12 is equal to zero. Used with register to register instructions.
3-5		Not Used.

Table 4-7
Register A Field CROM 12 through 12 (Sheet 2 of 2)

Value	Syntax	Detailed Definition
6	R(SO) or IR(SO) or BR(SO) or FPR(SO)	A-port address bits 01 and 02 are driven by I0 register bits 10 and 09 respectively. Address bit 00 is driven to a logical one and bit 03 is driven to a logical zero. Used to select an odd register for register to register doubleword instructions.
7	R(IR) or IR(IR) or BR(IR) or FPR(IR)	A-port address bits 00 through 03 are driven by I0 register bits 15, 14, 13, and CROM bit 12 respectively. CROM bit 12 is equal to zero.
8	R(BR) or IR(BR) or BR(BR) or FPR (BR)	A-port address bits 00 through 03 are driven by I0 register bits 08, 07, 06 and CROM bit 12 respectively. CROM bit 12 is equal to one. The ES field (CROM bits 20 through 23) is forced to equal three (I.FILE). Used only to address a base register.
9	R(BRO) or IR(BRO) or BR(BRO) or FRP(BRO)	A-port address bits 01 through 03 are driven by I0 register bits 07, 06 and CROM bit 12 respectively. Address bit 00 is driven to a logical one. CROM bit 12 is equal to one. The ES field (CROM bits 20 through 23) is forced to equal three (I.FILE). Used only to address a base register.
A	R(IBS) or IR(IBS) or BR(IBS) or FPR(IBS)	A-port address bits 00 through 03 are driven by I0 register bits 11, 10, 09, and CROM bit 12 respectively. CROM bit 12 is equal to one. The ES field (CROM bits 20 through 23) is forced to equal three (I.FILE). Used only to address a base register.
B-E		Not Used.
F	R(BB) or IR(BB) or BR(BB) or FPR(BB)	A-port address bits 00 through 03 are driven by I0. Register bits 15,14,13, and CROM bit 12 respectively CROM bit 12 is equal to one. The ES field (CROM bits 20 through 23) is forced to equal three (I.FILE). Used only to address a base register.

Table 4-8
Register B Field CROM 16 through 19 (Sheet 1 of 2)

Reference:	IE Unit logic drawing 130-103584, sheet 26
General:	If MB (CROM bit 32) = 0, the hexadecimal value of CROM bits 16 through 19 directly represents the B-port address of one of the 16 registers (0 through F) in the uP2901s, the IE Unit base register file, and the floating-point accelerator (FPA) file. If MB = 1, CROM bits 16 through 19 are used with a multiplexer which selects bits from the IMUX Bus and uses them to determine the B-port address of one of the 16 registers (0 through F) in the uP2901s, the IE Unit base register file, and the FPA file. Inputs to the multiplexer are detailed in the following table. File addresses are picked up at the end of the CROM cycle and used in the CREG cycle. If the instruction in the I1 register is moving into the I0 register (HADVANCEI0) during the CROM cycle, then the I1 register is the file address source; otherwise, the I0 register is the source. An RB field value can be an internal destination, an external destination, or an internal source, but never an external source. For internal and external destinations, the syntax appears to the left of the = sign; for internal sources, the syntax must appear to the right of the = sign and also to the right of an arithmetic operator in a double-operand expression. If the syntax appears on both sides of the =sign, the RB values inside both sets of parentheses must be equal. Either internal and external combination may be to the left of the = sign. For example, R(R)= is an internal destination, FPR(R),R(R)= is a combination of an external destination and an internal destination, and R(R), BR(R)= is a combination of an internal destination and an external destination. The IR ( ) syntax forces a value of 3, the BR ( ) forces a value of 7, and the FPR ( ) forces a value of B into the ED field (CROM bits 36 through 39). In the uP2901, the LSB is 00 and the MSB is 03.

Value	Syntax	Detailed Definition
0	R(R) or IR(R) or BR(R) or FPR(R)	B-port address bits 00 through 03 are driven by IMUX Bus bits 08, 07, 06 and CROM bit 16 respectively. CROM bit 16 is equal to zero. Used with arithmetic or logical add, subtract, multiply, divide, load and store instructions.

Table 4-8
Register B Field CROM 16 through 19 (Sheet 2 of 2)

Value	Syntax	Detailed Definition
1	R(RO) or IR(RO) BR(RO) FPR(RO)	B-port address bits 01 through 03 are driven by IMUX Bus bits 07, 06 and CROM bit 16 respectively. Address bit 00 is driven to a logical one. CROM bit 16 is equal to zero. Used to select an odd register during doubleword instructions.
2	R(S) or IR(S) or BR(S) or FPR(S)	B-port address bits 00 through 03 are driven by IMUX Bus bits 11, 10, 09, and CROM bit 16 respectively. CROM bit 16 is equal to zero. Used with register to register instructions.
3-7		Not Used.
8	R(BR) or IR(BR) or BR(BR) or FPR(BR)	B-port address bits 00 through 03 are driven by I0 register bits 08, 07, 06 and CROM bit 16 respectively. CROM bit 16 is equal to one. Used only to address a base register.
9	R(BRO) or IR(BRO) or BR(BRO) or FPR(BRO)	B-port address bits 01 through 03 are driven by I0 register bits 07, 06, and CROM bit 16 respectively. Address bit 00 is driven to a logical one. CROM bit 16 is equal to one. Used only to address a base register.
A	R(IBS) or IR(IBS) or BR(IBS) or FPR(IBS)	B-port address bits 00 through 03 are driven by I0 register bits 11, 10, 09, and CROM bit 16 respectively. respectively. CROM bit 16 is equal to one. Used only to address a base register.
B-F		Not Used.

Table 4-9
External Source Field CROM 20 through 23 (Sheet 1 of 5)

References:	MS Unit logic drawing 130-103568, sheet 39 CS Unit logic drawing 130-103583, sheet 39 IE Unit logic drawing 130-103584, sheet 30
General:	Provides input to the D-port of the uP2901, by means of the DB Bus, from sources detailed in the following table. For additional external sources, see External Source overlay, table 4-10.

Value	Syntax	Signal	Detailed Definition
0	CP.STATS	LCPUSTATOE	CPU status. The contents of the cache data out and I0 status registers are fed to bits 08 through 31 of the DB Bus. For definitions of CPU status word bits, see the CPU Status Word Indications table in Chapter 2.
1	I0	LI0BUSOE & LI0BUSSEOE	I0 Bus. The contents of the I0 register are fed to bits 00 through 31 of the DB Bus.
2	N.SHIFTER	LSHIFTEROE & LMDBOE	Nibble shifter. The contents of the nibble shifter are fed to bits 00 through 31 of the DB Bus. The N counter counts down by one.
3	IR( ) or BR( )	LBASETODBOE	Base file. The A-port output of one of BR( )sixteen dual-ported RAM registers is fed, via the base drivers, to bits 00 through 31 of the DB Bus. The address of the selected register is controlled by the RA field (table 4-7) of the microword and the value of the MA bit (CROM bit 24).

Table 4-9
External Source Field CROM 20 through 23 (Sheet 2 of 5)

Value	Syntax	Signal	Detailed Definition
4	CACHE. DOUT.MSW	LEVEHWSEOE, LEVEWORDOE & LMDBOE	Cache data out most-significant word. The contents of the cache data out even (MSW) register are fed to MDB Bus bits 00 through 31. The input to the register comes from the right shifter. CACHE.DOUT.MSW and CACHE.DOUT.LSW (external source value 5) must be used on the first microinstruction of a macroinstruction emulation sequence. After the first microinstruction, the cache data out registers may be changed by subsequent operand prefetching by the I unit portion of the IE unit. If the emulation sequence cannot comply with this restriction, then the operation code must lock LMAR when moving through the pipeline from the I2 register to the I1 register via the decode attribute mechanism. Standalone microcode may inhibit prefetching via the STOP.PREFETCH order (group 0, line 0), which stops both operand and instruction prefetching.
5	CACHE. DOUT.LSW	LODDWORDOE & LMDBOE	Cache data out least-significant word. The contents of the cache data out odd (LSW) register are fed to MDB Bus bits 00 through 31. The input to the register comes from the right shifter. See the definition of CACHE.DOUT.MSW (external source value 4) for parameters for the use of this source.
6	CACHE. DOUT.SE	LEVEHWSEOE, LMDBOE & LCACHESEOE	Cache data out sign extend. The contents of the cache data out sign-extend register are fed to MDB Bus bits 00 through 15 and cache data out even (MSW) register bits 16 through 31 are fed to MDB Bus bits 16 through 31. The input to the register comes from the right shifter.

Table 4-9
External Source Field CROM 20 through 23 (Sheet 3 of 5)

Value	Syntax	Signal	Detailed Definition
7	PSW	LPCOE & LMPCTODBOE	Program status word (PSW). The 4-bit 4-bit contents of the PSW status register are fed to DB Bus bits 00 and 05 through 07. The condition code logic feeds DB Bus bits 01 through 04. DB Bus bits 08 through 31 are fed from the macroprogram counter (MPC) via the MPC buffer. Bits 08 through 29 are the contents of the MPC, while bit 30 indicates that the next instruction is a right halfword. Bit 31 indicates that the last instruction was a right halfword.
8	BYTE CONSTANT	LDBSE0T7EN, LBYTESEOE, LBYTECTOE & LHWSEOE	Byte constant. A byte literal is fed from the P and C fields (CROM bits 52 through 59) of the microword to bits 24 through 31 of the DB Bus. The byte sign bit (CROM bit 52) is extended to fill DB Bus bits 00 through 23. This means that only numbers from decimal +127 to -128 or hexadecimal 0000007F to FFFFFF80 may be expressed.
9	LITERAL	LLITPROMOE	Literal PROM. A selected 32-bit literal is fed to DB Bus bits 00 through 31 from the 32-bit constant PROM. The PROM address is determined by the hexadecimal value of P and C fields of the microword (CROM bits 52 through 59). Any word from the 512 word by 32 bit literal file can be addressed by this source. The literal file can also be addressed by external source BITMASK.M (value E), the external source overlay BITMASK.R (value 0), and the S.L.H.BANK order (group 0,line A).

Table 4-9
External Source Field CROM 20 through 23 (Sheet 4 of 5)

Value	Syntax	Signal	Detailed Definition
A	EFMAR	LEFFMAREN	Effective memory address register. The contents of the logical MAR copy register are fed to DB Bus bits 00 through 04, 12, 30 and 31. The contents of the effective MAR are fed to DB Bus bits 05 through 11 and 13 through 29.
B	IO.SE	LDBSE0T7EN, LIOBUSSEOE, LHWSEOE & LIOHWSE LIOHWSE	I0 Bus sign extended. I0 register bits 16 through 31 are fed to DB Bus 16 through 31. Look-ahead mux bit 16 (the sign bit) is extended to feed DB Bus bits 00 through 15.
C	FPA.E	LAEEN	Floating-point arithmetic exception. The output of the add/subtract unit's arithmetic exception register is fed to DB Bus bits 00 through 07. The output of the multiply/divide unit's arithmetic exception register is fed to DB Bus bits 16 through 31.
D	FPR( )		Floating-point file. The A-port output of one of the sixteen dual-ported RAM registers in the floating-point file is fed to DB Bus bits 00 through 31. The address of the selected register is controlled by the RA field (table 4-7) of the microword and the value of the MA bit (CROM bit 24).
E	BITMASK.M	LBITMASKMEN, LBITMASKMOE & LLITPROMOE	Bit mask memory. A selected 32-bit literal is fed to DB Bus bits 00 through 31 from the 32-bit constant PROM. The PROM address is determined by I0 bits 06 through 08 and effective LMAR bits 30 and 31. Any of the first 32 words from the 512 word by 32 bit literal file can be addressed by this source. This source is used for bit in memory (e.g., TBM) type instructions. The literal file can also be addressed by external source LITERAL (value 9) and external source overlay BITMASK.R (value 0).

Table 4-9
External Source Field CROM 20 through 23 (Sheet 5 of 5)

Value	Syntax	Signal	Detailed Definition
F	BCK.PC	LDBSE0T7EN, LBYTESEOE, LHWSEOE, LBCKPCOE & HBCKPC	Backdate program counter. The 4-bit output of the backdate instruction counter is fed, via a buffer, to DB Bus bits 27 through 31. DB Bus bit 31 is held at zero, aligning the backdate program counter output to a halfword boundary. DB Bus bits 00 through 26 are filled with zeros. To backdate the macroprogram counter, the read backdate count (BCK.PC) must always precede the read PSW (PSW, external source value 7) by one cycle, unless the pipeline is frozen via an uncommitted branch mechanism. The BCK.PC must be coded no earlier than the second cycle following a FLUSH.I0 order (group 0, line 9). The BCK.PC CREG cycle may cause a one-cycle stopclock if a pipeline update PC count sequence is in progress. BCK.PC prevents pipeline changes during the CREG+1 cycle.

Table 4-10
External Source Overlay CROM 20 through 23 (Sheet 1 of 3)

References:	MS Unit logic drawing 130-103568, sheet 39 CS Unit logic drawing 130-103583, sheet 39 IE Unit logic drawing 130-103584, sheet 30
General:	Provides input to the D-port of the uP2901, by means of the DB Bus, from sources detailed in the following table. This field, enabled when the ES.OVL order (group 1, line A) is selected, replaces the ES field shown is table 4-10.

Value	Syntax	Signal	Detailed Definition
0	BITMASK.R	LBITMASKREN, LBITMASKROE & LLITPROMOE	Bit mask register. A selected 32-bit literal is fed to DB Bus bits 00 through 31 from the 32-bit constant PROM. The PROM address is determined by I0 bits 06 through 08, 14, and 15. Any of the first 32 words from the 512 word by 32 bit literal file can be addressed by this source. This source is used for bit in register (e.g., TBR) type instructions. The literal file can also be addressed by external sources LITERAL (value 9) and BITMASK.M (value E).
1	PMAR.COPY	LPMARCOPYOE	Physical memory address copy register. The contents of the PMAR copy register are fed to DB Bus bits 00 through 31. Bits 30 and 31 are the C bits, bits 08 through 29 are the physical address from the PMAR Bus, bit 07 is the LMAR F bit, and bits 00 through 06 are held at zero. The operation code using this function must lock LMAR or stop prefetching to prevent prefetch destruction of the address in the PMAR copy register.
2	MAP.MISS	LMAPMISSEN, LDBSE0T7EN & LMAPMISS-LMAROE	Map miss memory address register. The contents of the map miss MAR are fed to Bus bits 08 through 31. DB Bus bit 31 is held at zero, aligning the map miss MAR output to a halfword boundary. DB Bus bits 20 through 30 are fed the logical address bits 08 through 18 from the 3-way adder. DB Bus bits 00 through 19 are zero-filled. This source is valid only for operand addresses generated by the I unit portion of the IE unit. LMAR is automatically locked by a map miss microinterrupt.

Table 4-10
External Source Overlay CROM 20 through 23 (Sheet 2 of 3)

Value	Syntax	Signal	Detailed Definition
3	LMAR.COPY	LDBSE0T7EN, LMARCOPYEN, LLMARCOPY & LLMARCOPYOE	Logical memory address copy register. The contents of the LMAR copy register are fed to DB Bus bits 08 through 29 DB Bus bits 30 and 31 are fed by the C bit logic. Bits 00 through 07 are zero-filled. This source code is valid only for operand addresses generated by the I-unit portion of the IE unit. LMAR is automatically locked by a map miss microinterrupt. This source does not track the increment LMAR order (group 1, line 4). The operation code using this function must lock LMAR or stop prefetching to prevent prefetch destruction of the address in the LMAR copy register.
4	RAMDATA OUT.MSW	LMDBOE & LRAMDATA-OUTMSWOE	RAM data out most-significant word. RAMDATA Bus bits 00 through 31 (the most significant word) are fed to MDB Bus bits 00 through 31 via the RAM data out register. RAMDATA Bus bits 00 through 31 contain CROM bits 00 through 31 after an ACS, WCS, or PROM read.
5	RAMDATA OUT.LSW	LMDBOE & LRAMDATA-OUTLSWOE	RAM data out least-significant word. RAMDATA Bus bits 32 through 63 (the least-significant word) are fed to MDB Bus bits 00 through 31 via the RAM data out register. RAMDATA Bus bits 32 through 63 contain CROM bits 32 through 63 after an ACS, WCS, or PROM read.
6	N&IPLADDR	LNIPLOE & LMDBOE	N counter and initial program load address. The 16-bit IPL address is fed to DB Bus bits 16 through 31. The 8-bit output of the N counter is fed to MDB Bus bits 00 through 07. The input to the N counter is the YB Bus.
7	INT.LVL	LINTLVLOE & LMDBOE	Serial interrupt poll. The output of the serial interrupt poll circuit is fed to MDB Bus bits 08 through 15 and 21 through 29.

Table 4-10
External Source Overlay CROM 20 through 23 (Sheet 3 of 3)

Value	Syntax	Signal	Detailed Definition
8	SCRATCH	LSCRATCHOE & LMDBOE	Scratch pad. The contents of the selected scratch pad address are fed to DB Bus bits 00 through 31. The scratch pad can be addressed either by YB Bus bits 08 through 15 or by the P and C fields (CROM bits 52 through 59) of the microword. The scratch pad has 512 addressable locations. See the S.L.H. BANK order (group 0, Line A).
9			Not Used.
A	PCLATCH	LPCLATCHOE & LMDBOE	Microprogram counter latch. The contents of the uPC latch register are fed to MDB Bus bits 16 through 31.
B-D			Not Used.
E	MDB.BUS	LMDBOE	MDB.BUS Bits 00 through 31 of the MDB Bus are filled with ones and gated to bits 00 through 31 of the DB Bus. This source allows microdiagnostics to check for shorted bits.
F	DB.BUS		DB Bus. Bits 00 through 31 of the DB Bus are filled with ones. This source allows microdiagnostics to check for shorted bits.

Table 4-11
Internal Source Field CROM 24 through 27

Reference:	IE Unit logic drawing 130-103584, sheet 26
General:	CROM bit 24 = MA, the macro-A bit, used with the RA field (see table 4-7). The octal value of CROM bits 25 through 27 determines the sources, within the uP2901, of the two 4-bit ALU input words R and S. These words are used by the ALU as detailed in table 4-12.

Value	R Source	S Source
0	RAM A-port	Q register
1	RAM A-port	RAM B-port
2	0	Q register
3	0	RAM B-port
4	0	RAM A-port
5	Direct data input (DB Bus)	RAM A-port
6	Direct data input (DB Bus)	Q register
7	Direct data input (DB Bus)	0

Table 4-12
ALU Function Field CROM 28 through 31

Reference: IE Unit logic drawing 130-103584, sheet 26

General: CROM bit 28 = Cn, the carry-in bit, used in arithmetic operations only.

The octal value of CROM bits 29 through 31, along with the Cn bit, determines the arithmetic or logical operations which will be performed on the input words R and S by the ALU.

See table 4-11 for R and S sources and table 4-13 for the destination of the results of the ALU operation.

MULTIPLY (group 0, line D) and DIVIDE (group 1, line E) orders may modify the least-significant bit (bit 33) of the ID field.

Cn	Value	Syntax	Detailed Definition
0	0	R+S	R plus S
1	0	R+S+1	R plus S plus 1
0	1	-R+S-1	S minus R minus 1
1	1	-R+S	S minus R
0	2	R-S-1	R minus S minus 1
1	2	R-S	R minus S
X	3	R:S	R OR S
X	4	R&S	R AND S
X	5	%R&S	R-NOT AND S
X	6	R!S	R EX-OR S
X	7	%(R!S)	R EX-NOR S

Table 4-13
Internal Destination Field CROM 32 through 35

Reference:	IE Unit logic drawing 130-103584, sheet 26
General:	CROM bit 32 = MB, the macro-B bit, used with the RB field (see table 4-8). The octal value of CROM bits 33 through 35 determines the destination, within the uP2901, of the results of the ALU operation (see table 4-12). If CROM bit 33 = 1, the ALU results are shifted, right or left, before reaching the chosen destination. See the ALU shift field, table 4-23, for detailed shift descriptions.

Value	Syntax	Detailed Definition
0	Q	The ALU results are fed to the Q register and the Y Bus.
1	NOD	No internal destination. The ALU results are fed only to the Y Bus.
2	R(B)EX. DEST R(A)	The ALU results are fed to the RAM register chosen by the RB field (table 4-8). The contents of the RAM register chosen by the RA field (table 4-7) are fed to the Y Bus.
3	R(B)	The ALU results are fed to the RAM register chosen by the RB field (table 4-8).
4	R(B,SRZD)	The ALU results are shifted right and then fed to both the Q register and the RAM register chosen by the RB field (table 4-8). The ALU results are also fed, unshifted, to the Y Bus. See the SRAD, SRLD, and SRLRQ shifts in table 4-23.
5	R(B,SRX)	The ALU results are shifted right and then fed to the RAM register chosen by the RB field (table 4-8). The ALU results are also fed, unshifted, to the Y Bus. See the SRA, SRL, and SRC shifts in table 4-23.
6	R(B,SLZD)	The ALU results are shifted left and then fed to both the Q register and the RAM register chosen by the RB field (table 4-8). The ALU results are also fed, unshifted, to the Y Bus. See the SLAD, SLLD, and SLCRQ shifts in table 4-23.
7	R(B,SLX)	The ALU results are shifted left and then fed to the RAM register chosen by the RB field (table 4-8). The ALU results are also fed, unshifted, to the Y Bus. See the SLA, SLL, and SLC shifts in table 4-23.

Table 4-14
External Destination Field CROM 36 through 39 (Sheet 1 of 3)

References:	MS Unit logic drawing 130-103568, sheet 39 CS Unit logic drawing 130-103583, sheet 39 IE Unit logic drawing 130-103584, sheet 39
General:	Sends the Y port output of the uP2901, by means of the YB Bus, to destinations detailed in the following table. For additional external destinations, see External Destination overlay, table 4-16.

Value	Syntax	Signal	Detailed Definition
0	N.SHIFTER		Nibble shifter. YB Bus bits 00 through 31 are fed to the nibble shifter. The nibble shifter is always loaded regardless of which external destination is selected.
1	CMC	LLDCMCREG	Cache memory control. YB Bus bits 27 through 31 are loaded into the cache memory control register.
2	CACHE.DIN	LLDCACHDIN	Cache data in register. YB Bus bits 00 through 31 are loaded into the cache data in register.
3	IR( )	LWRBRFILE	Instruction file if B file address is not equal to zero. YB Bus bits 00 through 31 are written to a register specified by the B file address. If the B file address is equal to zero, this function is inhibited.
4	L.SHARED.MEM	LLDSHMREGL	Lower shared memory limit register. YB Bus bits 07 through 12 are loaded into the lower shared memory limit register.
5	STATS	LLDPSWSTAT	Program status word status register. YB bus bits 00 and 05 through 07 are loaded into the PSW status register. These bits indicate CPU status where: 00 = Privileged mode 05 = Extended addressing mode 06 = Base mode 07 = Arithmetic exception trap

Table 4-14
External Destination Field CROM 36 through 39 (Sheet 2 of 3)

Value	Syntax	Signal	Detailed Definition
6	PC	LLDPCFROMY	Macroprogram counter. YB Bus bits 09 through 30 are loaded into the macroprogram counter via the input source multiplexer.
7	BR()	LWRBRFILE	Base file. Y Bus bits 00 through 31 are written into a selected register in the base file dual-ported RAM. The register address is controlled by the RB field of the microword and the value of the MB bit (CROM bit 32).
8	MAP.HIT.REG	LLDMHITREG	Map hit register. YB Bus bits 16 through 31 are loaded into the map hit register.
9	U.SHARED.MEM	LLDSHMREGH	Upper shared memory limit register. YB Bus bits 07 through 12 are loaded into the upper shared memory limit register.
A	EMAR	LLDEMAR	Execution memory address register. YB Bus bits 00 through 31 are loaded into the execution memory address register.
B	FPR()	LCPUTOFILE & LWRBRFILE	Floating-point file. Y Bus bits 00 through 31 are written into a selected register in the dual-ported RAM of the floating-point accelerator. The register address is controlled by the RB field of the microword and the value of the MB bit (CROM bit 32). This destination also writes to the I file RAM if the B file address is not equal to zero
C	A/S.MSW		Add/subtract unit most-significant word. One word from the DB Bus and one word from a selected FPA file register are loaded into the two single-precision (MSW) operand input registers in the FPA's add and subtract unit. The register address is controlled by the RB field of the microword and the value of the MB bit (CROM bit 32).

Table 4-14
External Destination Field CROM 36 through 39 (Sheet 3 of 3)

Value	Syntax	Signal	Detailed Definition
D	A/S.LSW		Add/subtract unit least-significant word. One word from the DB Bus and one word from a selected FPA file register are loaded into the two double-precision (LSW) operand input registers in the FPA's add and subtract unit. The register address is controlled by the RB field of the microword and the value of the MB bit (CROM bit 32).
E	M/D.MSW		Multiply/divide unit most-significant word. One word from the DB bus and one word from a selected FPA file register are loaded into the two single-precision (MSW) operand input registers in the FPA's multiply and divide unit. The register address is controlled by the RB field of the microword and the value of the MB bit (CROM bit 32).
F	M/D.LSW		Multiply/divide unit least-significant word. One word from the DB bus and one word from a selected FPA file register are loaded into the two double-precision (LSW) operand input registers in the FPA's multiply and divide unit. The register address is controlled by the RB field of the microword and the value of the MB bit (CROM bit 32).

Table 4-15
External Destination Overlay CROM 36 through 39 (Sheet 1 of 3)

References:	MS Unit logic drawing 130-103568, sheet 39 CS Unit logic drawing 130-103583, sheet 39 IE Unit logic drawing 130-103584, sheet 40
General:	Sends the Y port output of the uP2901, by means of the YB Bus, to destinations detailed in the following table. This field, enabled when the ED.OVL order is selected, replaces the ED field shown in table 4-14.

Value	Syntax	Signal	Detailed Definition
0	NOD		No destination.
1	RAMDATA IN.MSW	LLDRAM-DATAINMSW	RAM data in most-significant word. YB Bus bits 00 through 31 are loaded into the RAM data in most-significant word register.
2	RAMDATA IN.LSW	LLDRAM-DATAINLSW	RAM data in least-significant word. YB bus bits 00 through 31 are loaded into the RAM data in least-significant word register.
3	RAM.ADDR	LLDRAMADDR	RAM address. YB Bus bits 16 through 31 are loaded into the RAM address register.
4	N	LLDNCTR	N counter. YB Bus bits 00 through 07 are loaded directly into the N counter.
5	SCRATCH. ADDR	LLDSCRATCH-ADDR	Scratchpad address. YB Bus bits 00 through 15 are loaded into the scratchpad address register. The register is enabled by the S.ADDR order (group 3, line 7).
6	N.L.ZERO. CNT	LLDNCTR-LEAD	Count number of nibbles of leading zeros. YB Bus bits 00 through 31 are fed to the leading/trailing zeros detection PROM, where the number of nibbles of leading zeros is counted. The 8-bit result is loaded into the N counter.

Table 4-15
External Destination Overlay CROM 36 through 39 (Sheet 2 of 3)

Value	Syntax	Signal	Detailed Definition
7	N.T.ZERO.CNT	LLDNCRT-TRAIL	Count number of nibbles of trailing zeros. YB Bus bits 00 through 31 are fed to the leading/trailing zeros detection PROM, where the number of nibbles of trailing zeros is counted. The 8-bit result is loaded into the N counter.
8-A			Not Used.
B	FPR( )	LCPUTOFILE LWRBRFILE	Floating-point file. Y Bus bits 00 through 31 are written into a selected register in the dual-ported RAM of the floating-point accelerator. The register address is controlled by the RB field of the microword and the value of the MB bit (CROM bit 32). This destination also writes to the I file RAM if the B file address is not equal to zero.
C	A/S.MSW		Add/subtract unit most-significant word. One word from the DB bus and one word from a selected FPA file register are loaded into the two single-precision (MSW) operand input registers in the FPA's add and subtract unit. The register address in controlled by the RB field of the microword and the value of the MB bit (CROM bit 32).
D	A/S.LSW		Add/subtract unit least-significant word. One word from the DB bus and one word from a selected FPA file register are loaded into the two double-precision (LSW) operand input registers in the FPA's add and subtract unit. The register address is controlled by the RB field of the microword and the value of the MB bit (CROM bit 32).

Table 4-15
External Destination Overlay CROM 36 through 39 (Sheet 3 of 3)

Value	Syntax	Signal	Detailed Definition
E	M/D.MSW		Multiply/divide unit most-significant word. One word from the DB bus and one word from a selected FPA file register are loaded into the two single-precision (MSW) operand input registers in the FPA's multiply and divide unit. The register address is controlled by the RB field of the microword and the value of the MB bit (CROM bit 32).
F	M/D.LSW		Multiply/divide unit least-significant word. One word from the DB bus and one word from a selected FPA file register are loaded into the two double-precision (LSW) operand input registers in the FPA's multiply and divide unit. The register address is controlled by the RB field of the microword and the value of the MB bit (CROM bit 32).

Table 4-16
Order Enable and Order Group Fields CROM 40 through 51

References:
MS Unit logic drawing 130-103568, sheet 40
IE Unit logic drawing 130-103584, sheet 32

General:

The binary value of CROM bits 40 through 43 determines the order groups to be enabled. CROM bits 40 through 43 are low/true; therefore, @F indicates that no order group is enabled. Any number of groups may be enabled at the same time.

CROM bit 40 = 0	Order group 0 enabled
CROM bit 41 = 0	Order group 1 enabled
CROM bit 42 = 0	Order group 2 enabled
CROM bit 43 = 0	Order group 3 enabled

The hexadecimal value of the order group 0/1 (CROM bits 44 through 47) and order group 2/3 (CROM bits 48 through 51) fields specify the individual lines within their enabled groups.

The two order group pairs, 0/1 and 2/3, operate independently. This means that the line chosen in group 0 need not agree with the line chosen in group 2/3.

Orders 8 through F of order groups 0 and 1 are CROM cycle orders. All other orders are CREG cycle orders.

For detailed definitions of individual orders see tables 4-17 through 4-20.

Table 4-17
Order Group 0 (Sheet 1 of 5)

Value	Syntax	Signal	Detailed Definition
0	STOP. PREFETCH	LSTOPPRE-FETCHORD	Stop prefetch. The output of this order is connected to the prefetch logic in the I unit portion of the IE unit. The order causes the I unit to immediately cease operand prefetching, and to cease instruction prefetching when both the I3 and I2 registers are full. The effects of this order are cleared by either the JUMPD sequence (value 7) or the FLUSH.PIPELINE order (group 3, line B).
1			Not Used.
2	TTYPE.OVL	LTTYPEOVLORD	Transfer type overlay. The output of this order is connected to the cache transfer type logc. The order replaces the SelBUS codes field (table 4-24) with the SelBUS overlay field (table 4-25). This order is used in conjunction with either the BUSREQ.E (group 3, line E) or the BUSREQ.L (group 3, line F) order.
3	I.RELEASE. LMAR	LIRELEAS-LMARORD	Release logical memory address register. The output of this order is connected to the load LMAR logic circuitry. The order is used by the firmware to release a hold which has previously been placed on the LMAR. This order is equivalent to the RELEASE.LMAR order (group 2, line D).

Table 4-17
Order Group 0 (Sheets 2 of 5)

Value	Syntax	Signal	Detailed Definition
4	CIRC.SHIFT	LCURRENTORD	Circular nibble shift. The output of this order is connected to the select inputs of the three 2:1 multi-plexers in the nibble shifter. The order is used in conjunction with either the RN.SHIFT (group 2, line 6) or the LN.SHIFT (group 2, line 5) order. With the RN.SHIFT order, YB Bus bits 00 through 31 are shifted right four bit positions. Bits 28 through 31, shifted beyond the bit 31 position, are entered in the bit 00 through 03 positions. With the LN.SHIFT order, YB Bus bits 00 through 31 are shifted left four bit positions. Bits 00 through 03, shifted beyond the bit 00 position, are entered in the bit 28 through 31 positions.
5	LS.MS.FILL	LLSNORD	Least-significant most-significant nibble fill. The output of this order is fed to the select input of the 2:1 multiplexer in the MS unit's Y Bus control. The order is used in conjunction with either the RN.SHIFT (group 2, line 6) or the LN.SHIFT (group 2, line 5) order. If the order is given, the least-significant YB Bus nibble of the previous nibble shift (bits 28 through 31) is fed to the nibble shifter; if not given, the most-significant nibble (bits 00 through 03) is fed. In the nibble shifter, a RN.SHIFT shifts the word four bits to the right, bits 28 through 31 are discarded, and bit positions 00 through 03 are filled with the least-significant bits of the previous nibble shift. A LN.SHIFT shifts the word four bits to the left, bits 00 through 03 are discarded, and bit positions 28 through 31 are filled with the most-significant bits of the previous nibble shift. This order provides two pass, double-precision nibble shift capability.

Table 4-17
Order Group 0 (Sheet 3 of 5)

Value	Syntax	Signal	Detailed Definition
6	ZERO.FILL	HZROFILLORD	Zero nibble fill. The output of this order is fed to the strobe inputs of the three 2:1 multiplexers in the nibble shifter. The order is used in conjunction with either the RN.SHIFT (group 2, line 6) or the LN.SHIFT (group 2, line 5) order. A RN.SHIFT shifts the word four bits to the right, bits 28 through 31 are discarded, and bit positions 00 through 03 are zero-filled. A LN.SHIFT shifts the word four bits to the left, bits 00 through 03 are discarded, and bit positions 28 through 31 are zero-filled.
7	SETCC1(X)	LSETCCORDL	Set condition codes. The output of this order is fed to the carry-save flip-flop and the condition code FPLAs. The order sets the condition codes according to the rules detailed in table 4-21. The (X) is a variable determined by the hexadecimal value of CROM bits 52 through 55. This order is equivalent to the SETCC(X) order (group 2, line F). This is a CREG order used in the CREG+1 cycle.
8	RDACS	LRDACSORD	Read alterable control store (ACS) RAM. This order, when given with the RAMRD order (group 1, line 8), causes the data at the ACS location addressed by the contents of the RAM address register to be loaded into the RAM data out MSW register and the RAM data out LSW register. This is a CROM order used in the CREG cycle.

Table 4-17
Order Group 0 (Sheet 4 of 5)

Value	Syntax	Signal	Detailed Definition
9	FLUSH.I0	LFLUSHI0ORD	Flush I0 register. The output of this order is gated to the backdate program counter, the I0 register, and the I0 status register. The order clears the I0 register, any error status pending against it, and decrements the backdate program counter. This is a CROM order used in the CREG cycle.
A	S.L.H.BANK	HHIIBANKORD & LHIBANKCREG	Scratchpad/literal high bank. The output of this order is fed, via CREG registers, to the scratchpad and 32-bit constant (literal) PROM. The order enables the CPU to address the high bank (@ 100 through 1FF) of the scratchpad and literal PROM. This is a CROM order used in the CREG cycle.
B			Not Used.
C	ED.OVL	LEDOVLORDL	External destination overlay. The output of this order is connected to the Y Bus control circuitry in the CS, IE, and MS units. The order replaces the external destination field (table 4-14) with the external destination overlay (table 4-15) when an external destination (CROM bits 36 through 39) is specified. This is a CROM order used in the CREG cycle.

Table 4-17
Order Group 0 (Sheet 5 of 5)

Value	Syntax	Signal	Detailed Definition
D	MULTIPLY	LMULTORD	Multiply. The output of this order is gated to the ALU shift multiplexer in the IE unit. This is a CROM cycle order in which the least-significant bit of the Q-register in the uP2901 is used to select an internal source R-S combination. The order is used in the multiply macroinstruction to produce either the R(A), R(B) combination, if the least-significant bit of the Q-register is one, or the 0, R(B) combination, if the least-significant Q-register bit is zero. This hardware assist provides a look at the multiplier to determine whether to do an add-and-shift or a shift only. On the clock prior to the MULTIPLY order, a single-precision shift to a dummy register must be performed to provide a look at the least-significant Q-register bit.
E	PUSH	LPUSHORD	Push uPC stack. The output of this order is gated to the S3 and S4 inputs of all four 4-bit-slice elements of the microprogram counter. The order causes the top level of the 4-word stack to be filled with the output of the full adder. Previous data residing in the top three words are pushed down one level in the stack and the bottom word is written over and lost.
F	POP	LPOPORD	Pop uPC stack. The output of this order is gated to the S3 input of all four 4-bit-slice elements of the microprogram counter. The order causes the words in the 4-word stack to be moved up one location. The top word in the stack is fed, through a 4:1 multiplexer, to the internal register of the uPC. The bottom word of the stack remains at that location as well as being pushed up to the third level.

Table 4-18
Order Group 1 (Sheet 1 of 5)

Value	Syntax	Signal	Detailed Definition
0	CLR29	LCLKLMAR-BUSYORD & LEUNITCLR29	Clear LMAR bit 29. The output of this order is fed, via error status circuit, to the LMAR counter in the CS unit. The order clears LMAR counter bit 29 and increments the macroprogram counter. This order is used to store the second word in a doubleword store operation.
1	UINTEN.A	LMICROINT-AENORD	Enable A microinterrupts. The output of this order is fed to the microinterrupt vector circuitry in the IE unit. The order enables both the A and B groups of microinterrupts. The A group includes the LINTEXCPINT, LADDRSTOPINT, and LAEXPINT signals. See the UINTEN.B (group 1, line 3) order for group B details.
2	EXECRH	HXCUTERHWORD	Execute right halfword. The output of this order is fed to the I3 status register. The order causes the I3 right halfword flag to become set when valid data is loaded into the I3 register from the CAMUX Bus. This order is used with the EXM right and EXPR macro instructions.
3	UINTEN.B	LMICROINT-BENORD	Enable B microinterrupts. The output of this order is fed to the microinterrupt vector circuitry in the IE unit. The order enables the B group of microinterrupts. This group includes the LDERRORINT, LDMAPMISSINT, and LAEXPINT signals. These interrupts are also enabled by the UINTEN.A (group 1, line 1) and INCLMAR. & UINTEN.B (group 1, line 5) orders.

Table 4-18
Order Group 1 (Sheet 2 of 5)

Value	Syntax	Signal	Detailed Definition
4	INCLMAR	LCLKLMAR-	Increment LMAR. The output of this order is gated to the LMAR counter in the CS unit. The order causes the contents of the LMAR counter to be incremented by one word.
5	INCLMAR. &UNITEN.B	LMICROINT-BENORD, LCLKLMAR-BUSYORD & LINCLMARORD	Increment LMAR and enable B microinterrupts. The output of this order is gated to the LMAR counter in the CS unit and fed to the micro interrupt vector circuitry in the IE unit. The order combines the functions of both the INCLMAR (group 1, line 4) and UINTEN.B (group 1, line 3) orders.
6	UINTEN.C	LMICROINT-CENORD	Enable C microinterupt. The output of this order is connected to the microinterrupt Vector circuitry. This order enables WCS microcode to detect and handle FPA arithmetic exceptions.
7	USECARRY	LUSECARRYORD	Use carry-out. The output of this order is fed to the set input of the use carry flip-flop in the IE unit. The order is used with doubleword arithmetic instructions. The order allows the carry-out from the first word of the instruction to be used as the carry-in to the second word of the instruction. This order must be coded with the first arithmetic operation of the doubleword pair and must be accompanied by a SETCC(D) order (see Table 4-21). The USECARRY function must be cleared by a subsequent SETCC(X) order without a USECARRY order. Normally, this occurs with the second arithmetic operation of the doubleword pair.

Table 4-18
Order Group 1 (Sheet 3 of 5)

Value	Syntax	Signal	Detailed Definition
8	RAMRD	LRAMRDORD	RAM read. The output of this order is gated to the RAM control circuitry in the MS unit. The order causes the read of a 64-bit microword from the control PROM the writable control store (WCS) RAM or, when given with the RDACS order (group 0, line 8), the alterable control store (ACS) RAM. The read data is strobed into the 64-bit RAM data out registers. The word is addressed from the RAM address register by use of the RAM.ADDR external destination (table 4-15, value 3). This is a CROM order used in the CREG cycle. The order causes a one-cycle stopclock during the CREG cycle. It should not be coded with SelBUS activity in progress.
9	INHLASTRHW	LINHLAST-RHWORD	Inhibit last right halfword. The output of this order is fed, via CREG register, to the macroprogram counter in the CS unit. The order inhibits the sensing of the right halfword portion of the previous macroinstruction. The order is used with branch and link macroinstructions. This is a CROM order used in the CREG cycle.
A	ES.OVL	LESOVLORD	External source overlay. The out-put of this order is connected to the DB and MDB Bus control circuitry in the MS, IE and CS units. The order replaces the external source field (table 4-9) with the external source overlay (table 4-15) when an external source (CROM bits 20 through 23) is specified.

Table 4-18
Order Group 1 (Sheet 4 of 5)

Value	Syntax	Signal	Detailed Definition
B			Not Used.
C	JUMP.DATA	LJUMPDATAORD	Jump data. The output of this order is fed to the microprogram counter (uPC) control circuit. The order will cause the uPC to use YB Bus bits 16 through 31 as the source of the next uPC address. This order is used for computed branches in firmware. Bits 00 through 03 of the microword S field must be coded with a long branch (value 3).
D	DCNTISAVE	LDNCNTI-SAVEORD	Decrement I0 register shift counter. The output of this order is connected to the enable inputs of the I0 register shift counter. When a shift instruction is issued, bits 11 through 15 (the shift field) of the I1 register are loaded into the I0 register shift counter. This order causes the contents of the register to be decremented by one with each clock cycle. The contents of the counter are checked after each shift by the ISAVE0 test (group 0, line B).

Sheet 4-18
Order Group 1 (Sheet 5 of 5)

Value	Syntax	Signal	Detailed Definition
E	DIVIDE	LDIVORD	Divide. The output of this order is gated to the ALU shift multiplexer in the IE unit. This is a CROM cycle order which used the equal-to-zero and sign outputs of the uP2901 to select an internal source R-S combination. When the FE0 signal is high or low and the HSIGN signal is low, the R(A), R(B) combination is selected and a subtract-and-shift occurs. When the FE0 signal is low and the HSIGN signal is high, the 0, R(B) combination is selected and only a shift function occurs. On the clock prior to the divide order, a subtract is done to compare the divisor and dividend. This comparison generates the HFE0 and HSIGN signals for the divide order. When the subtract-and-shift function is performed, based on the comparison of divisor and dividend, a one is shifted into the least-significant bit of the Q-register. When only a shift is performed, a zero is shifted into the least-significant bit of the Q-register. This is how the quotient is formed.
F	RAMWR	LRAMWRORD	RAM write. The output of this order is fed to the RAM control circuitry in the MS unit. The order, when level order 6 (RAM.WREN) is set, causes the data from the 64-bit RAM data in registers to be written to the WCS or ACS RAM. The RAM location is addressed by RAMADDR bus bits 00 through 15. This is a CROM order used in the CREG cycle. The order causes a one-cycle stopclock during the CREG cycle. It should not be coded with SelBUS activity in progress.

Table 4-19
Order Group 2 (Sheet 1 of 6)

Value	Syntax	Signal	Detailed Definition
0	LATCH.ORD	LLATCHORD	Latch level order. The output of this order is connected to the level order latching circuits in the MS and IE units. When given by itself, this order sets one of the orders from the level order select field (table 4-22) as determined by the hexadecimal value of CREG bits 52 through 55. When given with the LATCH.DATA order (group 3, line 0), the order will reset a previously set level order. A level order can be directly set or reset by the special syntax SET(X) or RESET(X), where X is the name of the level order.
1	CNTENORD	LCNTENORD	Enable N counter. The output of this order is gated to the enable input of the N counter. This order enables the N counter to count. If the DNCNTORD order (group 3, line 1) is also given, or the N counter selected as an external source, the count will be in the down direction; otherwise, the count will be upward. See the N.NOVR test (group 1, line 5).
2	UPACK	LUPACKORD	User panel acknowledge. The output of this order is routed, via the turnkey interface circuit, to the turnkey panel. The order is used to clear panel conditions. For every signal received by the CPU from the panel (excluding SelBUS communications), the CPU will respond with 'UPACK'.

Table 4-19
Order Group 2 (Sheet 2 of 6)

Value	Syntax	Signal	Detailed Definition
3	RSTINTEN	LRSTINTENORD	Reset interrupt enable flip-flops. The output of this order is connected to the clear inputs of the interrupt enable flip-flops in the MS unit. The order causes interrupts to be disabled. This condition will remain until interrupts are set by the SETINTEN order (group 2, line 4). The set output of the interrupt enable flip-flops is checked in the test structure by the INTRENA test (group 2, line 7). Interrupts are automatically re-enabled when the next valid macroinstruction is executed and completed. This order is normally used with uninterruptable pairs of macroinstructions. For this functionality, the order must be coded one cycle ahead of the JUMPD sequence (value 7).
4	SETINTEN	LSETINTENORD	Set interrupt enable flip-flops. The output of this order is connected to the preset inputs of the interrupt enable flip-flops in the MS unit. The order causes interrupts to be enabled. This condition will remain until interrupts are reset by the RSTINTEN order (group 2, line 3). The set output of the interrupt enable flip-flops is checked in the test structure by the INTRENA test (group 2, line 7).

Table 4-19
Order Group 2 (Sheet 3 of 6)

Value	Syntax	Signal	Detailed Definition
5	LN.SHIFT	LLFSHFORD	Left nibble shift. The output of this order is connected to the nibble shifter. The order can be given either by itself or in conjunction with any one of the CIRC.SHIFT (group 0, line 4), LS.MS.FILL (group 0, line 5), or ZERO.FILL (group 0, line 6) orders. The order causes the nibble shifter to shift YB Bus bits 00 through 31 four bit positions to the left. Bits 00 through 03 are either discarded or, with the CIRC.SHIFT order, shifted to the bit 28 through 31 positions. Bit positions 28 through 31 are either filled according to the definitions of the CIRC.SHIFT, LS.MS.FILL, or ZERO,FILL orders or, if no additional shift order is given, with the most-significant YB Bus nibble (bits 00 through 03). On each nibble shift, YB Bus bits 00 through 03 and 28 through 31 are saved in the LS MS fill registers for a subsequent nibble shift with an LS.MS.FILL order (group 0, line 5). Each nibble shift causes the N counter to automatically decrement by one.
6	RN.SHIFT	LRISHFORD	Right nibble shift. The output of this order is connected to the nibble shifter. The order can be given either by itself or in conjunction with any one of the CIRC.SHIFT (group 0, line 4), LS.MS.FILL (group 0, line 5), or ZERO.FILL (group 0, line 6) orders. The order causes the nibble shifter to shift YB Bus bits 00 through 31 four bit positions to the right. Bits 28 through 31 are either discarded or, with the CIRC.SHIFT order, shifted to the bit 00 through 03 positions. Bit positions 00 through 03 are either filled according to the definitions of the CIRC.SHIFT, LS.MS.FILL, or ZERO.FILL orders or, if no additional shift order is given, with the most-significant YB Bus nibble (bits 00 through 03). On each nibble shift, YB Bus bits 00 through 03 and 28 through 31 are saved in the LS MS fill registers for a subsequent nibble shift with an LS.MS.FILL order (group 0, line 5). Each nibble shift causes the N counter to automatically decrement by one.

Table 4-19
Order Group 2 (Sheet 4 of 6)

Value	Syntax	Signal	Detailed Definition
7	SCR.WRITE	LSCRATCH-WRORD	Scratchpad write. The output of this order is fed to the write-enable inputs of the scratchpad RAMs and the output-enable inputs of the nibble shifter. The order enables YB Bus bits 00 through 31, after passing through the nibble shifter and MDB Bus, to be written into the scratchpad. The scratchpad is addressed either by the contents of the microword's P and C fields (CROM bits 52 through 59) or by YB Bus bits 08 through 15 when the S.ADDR order is given and the SCRATCH.ADDR external destination is chosen.
8			Not Used.
9	RSTADDR-STOP		Reset address stop. This order clears pending address stops before a halt is executed.
A	FPA.A	LFPAAORD	Floating-point accelerator order A. The output of this order is fed to the FPA. This order is used in conjunction with the FPA.B order (group 3, line A) to direct either the add/subtract unit to add or subtract, or the multiply/divide unit to multipy or divide. In order to work properly, the orders must accompany, on the same clock, a load of either the add/subtract unit or the multiply/divide unit, as determined by the external destination (see tables 4-15 and 4-16, values C through F). These orders are given by use of special syntax statements. FP.ADD or FP.MPY gives both the FPA.A and FPA.B orders. FP.SUB or FP.DIV gives only the FPA.A order. The multiply/divide unit also does an integer multiply by use of the FIXED.MPY syntax statement, which will given the FPA.B order only. If the floating-point macroinstruction is double precision, two clock cycles are required. On the first clock, the add/subtract or multiply/divide unit is loaded with the least-significant word. On the second clock, the most-significant word is loaded and the FP.ADD, FP.SUB, FP.MPY, or F.DIV syntax is given.

Table 4-19
Order Group 2 (Sheet 5 of 6)

Value	Syntax	Signal	Detailed Defintion
B	BU	LBRNCHAL-WAYSORD	Branch unconditional. The output of this order is connected to the CPU stopclock circuitry in the IE unit. The order causes the macroprogram counter to be loaded with the contents of the LMAR counter register and the CPU to unconditionally branch to that memory location. This order is used with branch unconditional and branch and link macroinstructions.
C	TEST.CC	LTESTCC-BRNCHORD	Test condition codes. The output of this order is connected to the CPU stopclock circuitry in the IE unit. The order causes the condition codes test signal (HCCTEST), qualified by the HCREG00 signal, to be tested. If HCREG00=1 (BCF) and HCCTEST is false (0) or if HCREG00=0 (BCT) and HCCTEST is true (1), then the macroprogram counter is loaded with the contents of the LMAR counter register and the CPU branches (macro branch) to that memory location. If HCREG00=0 (BCT) and HCCTEST is false (0) or if HCREG00=1 (BCF) and HCCTEST is true, then the branch is refused and the CPU continues. For test details, see test group 3, line 6 (table 4-6).
D	RELEASE.LMAR	LIRELEAS-LMARORD	Release logical memory address register. The output of this order is connected to the load LMAR logic circuitry. The order is used by the firmware to release a hold which has previously been placed on the LMAR. This order is equivalent to the I.RELEASE.LMAR order (group 0, line 3).

Table 4-19
Order Group 2 (Sheet 6 of 6)

Value	Syntax	Signal	Detailed Definition
E	SETAEXP	LSETAEXPORD	Set arithmetic exception trap. The output of this order is gated to the set input of the arithmetic exception flip-flop in the IE unit. The order sets the arithmetic exception flip-flop, giving a positive AEXP test (group 3, line 3) result. If arithmetic exceptions and A microinterrupts are enabled, the arithmetic exception trap is set. The trap is reset by the RSTAEXP order (group 3, line D).
F	SETCC(X)	LSETCCORDL	Set condition codes. The output of this order is fed to the carry-save flip-flop and the condition code FPLAs. The order sets the condition codes according to the rules detailed in table 4-21. The (X) is a variable determined by the hexadecimal value of CROM bits 52 through 55. This order is equivalent to the SETCC1(X) order (group 0, line 7). This is a CREG order used in the CREG +1 cycle.

Table 4-20
Order Group 3 (Sheet 1 of 4)

Value	Syntax	Signal	Detailed Definition
0	LATCH.DATA	LLATCH-DATAORD	Latch order data. The output of this order is connected to the data inputs of the level order latches in the MS and IE units. This order, when given with the LATCH.ORD order (group 2, line 0), resets a previously set level order from the level order select field (table 4-22). A level order may be directly reset by the special syntax RESET(X), where X is the name of the level order.
1	DNCNTORD	LDNCNTORD	Decrement N counter. The output of this order is gated to the up/down input of the N counter. The order, when the CNTENORD order (group 2, line 1) is given, causes the N counter to count downward. If this order is used by itself (no CNTENORD), the N.NOVR test (group 1, line 5) indicates whether the N counter contents are equal to zero or not equal to zero. If this order is not given, the N.NOVR test indicates whether the N contents are equal to @FF or not equal to @FF.
2	HALT	LHALTORD	Halt. The output of this order is gated to the clear input of the run/halt flip-flop in the turnkey panel interface. The order resets the run/halt flip-flop, lighting the HALT LED and giving a positive result to the HALT test (group 0, line 4). The run/halt flip-flop is set by the RUN order (group 3, line 4) or by the control panel function.
3	RSTIPUTRAP	LRSTIPU-TRAPORD	Reset IPU trap. The output of this order is gated to the reset input of the IPU trap flip-flop in the MS unit. The order resets a previously set IPU trap.
4	RUN	LRUNORD	Run. The output of this order is fed to the preset input of the run/halt flip-flop in the turnkey interface panel. The order sets the run/halt flip-flop, lighting the RUN LED on the turnkey panel and giving a condition of false to the HALT test (group 0, line 4). The run/halt flip-flop is reset by the HALT order (group 3, line 2) or by the control panel function.

Table 4-20
Order Group 3 (Sheet 2 of 4)

Value	Syntax	Signal	Detailed Definition
5	STOP.EVENT	LSTOPEVENTORD	Stop event. The output of this order is connected to a counter in the Development Support System (DSS) or the Instrumentation Interface unit (IIU). If, while in the DSS mode, the microprogram counter (uPC) reaches @DE0 (invalid instruction), this order arms the DSS counter. If, after a count of 256 cycles, the uPC is still at @DE0, control is passed back to the CPU DSS or IIU. This order is used as a troubleshooting aid.
6			Not Used.
7	S.ADDR	LSCRATCH-ADDRORD	Scratchpad address. The output of this order connects, directly and via an inverter, to the output-enable inputs of the two scratchpad address register flip-flops. When given during a scratchpad operation, the order causes YB Bus bits 08 through 15 to be used as the source of the scratchpad address. If, during a scratchpad operation, the order is not given, the scratchpad address is determined by the hexadecimal value of the microword's P and C fields (CROM bits 52 through 59).
8			Not Used.
9	CLEAR.ERROR	LCLEARERROR	Clear errors. The output of this order is gated to flip-flops in the SelBUS tag and error status circuits. The order sets the ready signal and clears busy, retry, data fetch no transfer acknowledge, data write no transfer acknowledge, and instruction address stop errors. This order gives positive results to the READY (group 1, line A) test, and negative results to this IADDRSTOP (group 0, line 3), D.NOTA (group 1, line 2), D.NOTA1 (group 1, line 4), RETRY (group 2, line 2),W.NOTA (group 2, line 5), and BUSY (group 3, line 2) tests.

Table 4-20
Order Group 3 (Sheet 3 of 4)

Value	Syntax	Signal	Detailed Definition
A	FPA.B	LFPABORD	Floating-point accelerator order B. The output of this order is fed to the FPA. This order is used in conjunction with the FPA.A order (group 2, line A) to direct either the add/subtract unit to add or subtract, or the multiply/-divide unit to multiply or divide. For a more detailed description of this syntax, see the definition of the FPA.A order on table 4-19.
B	FLUSH. PIPELINE	HFLUSHPIPE, LFLUSHPIPE1 & LFLUSHPIPE2	Flush pipeline. The output of this order is fed to locations in the IE, CS, and MS units, including pipeline status registers and the backdate program counter. The order marks all instruction registers and their status invalid, and clears the backdate program counter.
C	TEST.ALUZ	LTESTALUZ-BRHORD	Test if ALU result is equal to zero. The output of this order is connected to the CPU stopclock circuitry in the IE unit. The order causes the ALU result equals zero signal (HFE0L), qualified by the HCREG00 signal, to be tested. If HCREG00=1 (BCF) and HFE0L is false(0) or if HCREG00=0 (BCT) and HFE0L is true(1), then the macroprogram counter is loaded with the contents of the LMAR counter register and the CPU branches (macro branch) to that memory location. If HCREG00=0(BCT) and HFE0L is false(0) or if HCREG00=1(BCF) and HFE0L is true, then the branch is refused and the CPU continues.
D	RSTAEXP	LCLRAEXPORD	Reset arithmetic exception trap. The output of this order is gated to the reset input of the arithmetic exception flip-flop in the IE unit. The order resets the arithmetic exception flip-flop, which, in turn, resets the arithmetic exception trap. The trap is set by the SETAEXP order (group 2, line E).

Table 4-20
Order Group 3 (Sheet 4 of 4)

Value	Syntax	Signal	Detailed Definition
E	BUSREQ.E	LEUNITAREQORD	EMAR SelBUS request. The output of this order is connected to the cache transfer-type logic in the IE unit. The order causes the execution memory address register (EMAR) to be used for a SelBUS transaction. The type of transfer is determined by the hexadecimal value of CROM bits 56 through 59. The transfer type is detailed in either the SelBUS codes field (table 4-24) or, if the TTYPE.OVL order (group 0, line 2) is also given, the SelBUS overlay field (table 4-25). This is a CREG order used by the CS unit in the CREG+1 cycle. If the CS unit does not accept the transaction, a CREG+1 stop clock results. IE unit prefetching must be stopped when this order is used.
F	BUSREQ.L	LEUNITBREQORD	LMAR SelBUS request. The output of this order is connected to the cache transfer-type logic in the IE unit. The order causes the logical memory address register (LMAR) to be used for a SelBUS transaction. The type of transfer is determined by the hexadecimal value of CROM bits 56 through 59. The transfer type is detailed in either the SelBUS codes field (table 4-24) or, if the TTYPE.OVL order (group 0, line 2) is also given, the SelBUS overlay field (table 4-25). This is a CREG order used by the CS unit in the CREG+1 cycle. If the CS unit does not accept the transaction, a CREG+1 stop clock results. IE unit prefetching must be stopped when this order is used.

Table 4-21
Condition Code Select Field CROM 52 through 55

Reference: IE Unit logic drawing 130-103584, sheet 33

General: This field sets the condition codes according to the rules in the table below. The hexadecimal value of CROM bit 52 through 55 determines the rules to be applied. The field is enabled when the SETCC(X) (group 2, line F) or SETCC1(X) (group 0, line 7) CREG order is given. CCs are set during the CREG+1 cycle from latched ALU results.

Legend:

GT = Greater than
LT = less than
ET = Equal to
AO= Arithmetic overflow
SO = Shift overflow

+ = ALU results are positive
- = ALU results are negative
0 = ALU results equal zero
NU = Not used (always zero)
EWL= Effective word length

* = To set CC4 for a doubleword operation, the LSW flag and LSW=0 flag must both be set. The MSW of the doubleword pair is equal to zero.

Value	Syntax	CC1	CC2	CC3	CC4	Comments
0	SETCC(A)	AO	+	-	0,*	Arithmetic.
1	SETCC(AM)	Old CC1	+	-	0,*	Arithmetic masked.
2	SETCC(L)	NU	+	-	0,*	Logical.
3	SETCC(D)	AO	+	-	0	Least-significant word of a doubleword pair.
4	SETCC(BIT)	Old bit	Old CC1	Old CC2	Old CC3	Used with bit in memory and bit in register instructions.
5	SETCC(Y)	Y01	Y02	Y03	Y04	Set CC's from the Y Bus.
6	SETCC(M)	NU	NU	NU	0,*	Used with compare masked instructions
7	SETCC (LSHIFT)	SO	NU	NU	NU	Left arithmetic shift instructions
8	SETCC(CA)	NU	GT	LT	ET	Compare arithmetic. The CC's show whether the register contents are greater than, less than or equal to the memory operand.
9	SETCC (INDIR)	EWL1 =1	EWL2 =1	EWL3 =1	EWL4 =1	Indirect. CC's are set equal to the state of bits 01 through 04 of the contents of the indirect word addressed in memory.
A-C						Not Used.

Table 4-22
Level Order Select Field CROM 52 through 55 (Sheet 1 of 3)

References: MS Unit logic drawing 130-103568, sheet 40
IE Unit logic drawing 130-103584, sheet 14

General: The hexadecimal value of CROM bits 52 through 56 determines the level order to be selected from the table below. Level orders are set by the order LATCH.ORDER (group 2, line 0). Level orders are reset by giving the orders LATCH.ORDER and LATCH.DATA (group 3, line 0) simultaneously. Level orders may be directly set or reset by the special syntax SET(X) or RESET(X), where X is the name of the level order.

Value	Syntax	Signal	Detailed Definition
0	PERR	HPERRORD	Parity error. The output of this order is connected to the turnkey panel interface. The order is used to drive the PE indication on the IOP console CRT.
1	INTACT	HINTACTORD	Interrupt active. The output of this order is connected to the turnkey panel interface. The order is used to drive the INTRPT LED on the turnkey panel and the ACT indication on the IOP console CRT.
2	WAIT	HWAITORD	Wait. The output of this order is connected to the turnkey panel interface. The order is used to drive the WAIT LED on the turnkey panel and the WAIT indication on the IOP console CRT.
3	DIS.PWRF	HDISPWRFORD	Disable powerfail. The output of this order is connected to the powerfail detection logic. When set, the order prevents the test structure and I2 exception vector circuitry from detecting a power failure.
4	DIS.UPATTN	HDISATTNORD	Disable user panel attention. The output of this order is connected to the turnkey panel interface. When set, the order prevents the I2 exception vector circuitry from detecting an attention request from the user panel.
5	CRAM	HCRAMORD	Control RAM. The output of this order is connected to the PROM and RAM enable and RAM control circuitry. When the order is set, the CPU is controlled by the microprogram from the alterable control store (ACS) instead of the PROM.

Table 4-22
Level Order Select Field CROM 52 through 55 (Sheet 2 of 3)

Value	Syntax	Signal	Detailed Definition
6	RAM.WREN	HRAMWREN ORD	RAM write enable. The output of this order is connected to the RAM write control circuitry. When set, the order enables the CPU to write into the writable control store (WCS) or the ACS when the RAMWR order (group 0, line 8) is given.
7	SIPU	HSIPUORD	Signal IPU. The output of this order is connected to the IPU trap circuitry and the SelBUS. When given, the order sets the IPU trap flip-flop in the MS unit of another processor.
8	FLAG	HFLAGL	Flag, latched. The output of this order is connected to the test structure (group 0, line 8). When set, the order indicates that the context of the CPU has been temporarily altered and must be restored before processing is resumed.
9	ERR.FLAG	HERR-FLAGL	Error flag, latched. The output of this order is connected to the test structure (group 1, line 1). When set, the order is used by subroutines to indicate that an error (memory or I/O) has occurred during subroutine execution. Status describing the error will be provided in a dedicated register.
A	ENAUTOMINT	HENAUTO-MINTL	Enable automatic microinterrupt, latched. The output of this order is connected to the I1 status register.
B	FPA		Floating-point accelerator. This order puts the floating-point accelerator on-line or off-line. When the order is set, the FPA is on-line; when reset, the FPA is off-line.
C	MULTI.CYCLE		Multiple-cycle instruction. The output of this order is connected to the test structure (group 2, line 6). When set, the order indicates that a multiple-cycle memory reference instruction is being executed.

Table 4-22
Level Order Select Field CROM 52 through 55 (Sheet 3 of 3)

Value	Syntax	Signal	Detailed Definition
D	MAP	HMAPL	Mapping mode, latched. The output of this order is connected to the test structure (group 3, line 1) and the map circuitry in the CS unit. When set, the order enables operation in the mapped mode.
E	TRACE	HTRACEL	Trace, latched. The output of this order is connected to the test structure (group 0, line 7) and the I3 status register in the IE unit. When set, the order enables the CPU to operate in the instruction-step (single-step) mode.
F	UNBLOCK	HUNBLOCK	Unblock interrupts. The output of this order is connected to the test structure (group 0, line C) and the I2 exception vector circuitry in the MS unit. When set, the order disables blocking of external interrupt requests. The order is controlled by software through use of the BEI (block external interrupts) and UEI (unblock external interrupts) macroinstructions and PSD2 bits 16 and 17.

Table 4-23
ALU Shift Code Field CROM 54 and 55 (Sheet 1 of 2)

Reference:	IE Unit logic drawing 130-103584, sheet 29
Influenced by:	CROM bits 33 through 35 of the ID field
General:	The ALU shift code field is enabled when CROM bit 33 of the ID field is equal to 1

Value	Syntax	Detailed Definition
0	SRA	Shift right arithmetic. All RAM bits are shifted right one bit position. A bit shifted beyond the bit 31 (LSB) position is truncated. The sign bit (bit 00) is extended right to fill the bit 01 positon.
	SRAD	Shift right arithmetic doubleword. All RAM and Q register bits are shifted right one bit position. A Q register bit shifted beyond the bit 31 (LSB) position is truncated. A RAM bit shifted beyond the bit 31 position enters the Q register at the bit 00 (MSB) position. The RAM sign bit (bit 00) is extended right to fill the RAM bit 01 position.
	SLA	Shift left arithmetic. The sign bit (bit 00) remains unchanged. RAM bits 01 through 31 are shifted left one bit position. A bit shifted beyond the bit 01 position is truncated. A zero is entered at the bit 31 (LSB) position.
1	SRL	Shift right logical. All RAM bits are shifted right one bit position. A bit shifted beyond the bit 31 (LSB) position is truncated. A zero is entered at the bit 00 (MSB) position.
	SRLD	Shift right logical doubleword. All RAM and Q register bits are shifted right one bit position. A Q register bit shifted beyond the bit 31 (LSB) position is truncated. A RAM bit shifted beyond the bit 31 position enters the Q register at the bit 00 (MSB) position. A zero is entered at the RAM bit 00 position.
	SLL	Shift left logical. All RAM bits are shifted left one bit position. A bit shifted beyond the bit 00 (MSB) position is truncated. A zero is entered at the bit 31 (LSB) position.

Table 4-23
ALU Shift Code Field CROM 54 and 55 (Sheet 2 of 2)

Value	Syntax	Detailed Definition
2	SRC	Shift right circular. All RAM bits are shifted right one bit position. A bit shifted beyond the bit 31 (LSB) position is entered at the bit 00 (MSB) position.
	SLC	Shift left circular. All RAM bits are shifted left one bit position. A bit shifted beyond the bit 00 (MSB) position is entered at the bit 31 (LSB) position.
	SLCRQ	Shift left circular RAM and Q register. All RAM and Q register bits are shifted left one bit position. A Q register bit shifted beyond the bit 00 (MSB) position is truncated. A RAM bit shifted beyond the bit 00 position enters both the RAM and Q registers at the bit 31 (LSB) position.
3	SRLRQ	Shift right logical RAM and Q register. All RAM and Q register bits are shifted right one bit position. Q register bits and RAM bits shifted beyond the bit 31 (LSB) position are truncated. A zero is entered at both the RAM and Q register bit 00 (MSB) positions.
	SLAD	Shift left arithmetic doubleword. The RAM sign bit (bit 00) remains unchanged. RAM bits 01 through 31 and all Q register bits are shifted left one bit position. A RAM bit shifted beyond the bit 01 position is truncated. A Q register bit shifted beyond the bit 00 (MSB) position enters the RAM at the bit 31 (LSB) position. A zero is entered at the Q register bit 31 position.
	SLLD	Shift left logical doubleword. All RAM and Q register bits are shifted left one bit position. A RAM bit shifted beyond the bit 00 (MSB) position is truncated. A Q register bit shifted beyond the bit 00 position enters the RAM at the bit 31 (LSB) position. A zero is entered at the Q register bit 31 position.

Table 4-24
SelBUS Codes Field CROM 56 through 59 (Sheet 1 of 3)

Reference:	IE Unit logic drawing 130-103584, sheet 11
General:	The SelBUS codes field is enabled by either the BUSREQ.E order (group 3, line E) or the BUSREQ.L order (group 3, line F). The hexadecimal value of CROM bits 56 through 59 determines the SelBUS transfer type selected from the table below. Refer to the SelBUS overlay, Table 4-25, for additional SelBUS codes.

Value	Syntax	Detailed Definition
0	IND.DREAD	Indirect data read. The IND.DREAD transfer is a CPU data read from main memory or cache similar to the DREAD (SelBUS code 6). It differs from the DREAD in that the returned data is fed via the CAMUX bus, to the I2 indirect register in the IE unit instead of the cache data out even (MSW) register. This transfer is used only by the I unit portion and not the E unit portion of the IE unit.
1	ODD.DREAD	Odd data read. The ODD.DREAD transfer is a CPU data read from main memory or cache similar to the DREAD (SelBUS code 6). It differs from the DREAD in that the returned data is fed, via the CAMUX bus, to the cache data out odd (LSW) register instead of the cache data out even (MSW) register.
2	IREAD	Instruction read. The IREAD transfer is a CPU instruction read from main memory or cache similar to the DREAD (SelBUS code 6). It differs from the DREAD in that instructions, rather than data, are being read. Also, the returned instructions are fed, via the CAMUX bus, to the I3 registers in the IE and MS units instead of the cache data out even (MSW) register.
3	CHK.STORE	Check map for store errors. The CHK.STORE transfer checks a map location (map image descriptor or MID) to determine if map errors (map invalid, map miss, or map protect) will occur if a map store is attempted. This transfer does not write to or read from cache or main memory.
4	READ.CHK.STORE	Read memory and check map for store errors. The READ.CHK.STORE transfer reads a location in main memory or cache. The transfer also checks the same location in the map (MID) to determine if errors (map invalid, map miss, or map protect) will occur if a memory and cache map store is attempted.

Table 4-24
SelBUS Codes Field CROM 56 through 59 (Sheet 2 of 3)

Value	Syntax	Detailed Definition
5	MAP	Map. The MAP transfer causes a logical to physical address operation and detects any resulting map errors (map miss or map invalid). The effective physical address is loaded into the physical memory address register (PMAR).
6	DREAD	Data read. The DREAD transfer is a CPU data read from the cache or main memory. The memory address from the physical memory address register (PMAR) bus is sent to the cache and, via the destination bus, to main memory. If the required memory location is found in the cache, and the cache information is valid, the main memory read is aborted; otherwise, the read is from main memory. If the cache is valid, the data is fed from the cache to the CAMUX Bus. If main memory is used, the data returns via the data bus to the CAMUX Bus. At the same time, the CS unit issues a DRT.WRITE (SelBUS overlay code 5) transfer, which updates the cache with the new memory location and data. From the CAMUX Bus, the data is fed to the cache data out even (MSW) register in the MS unit.
7	NOP.REQ	No operation request. The NOP.REQ transfer performs no SelBUS operation. This transfer is issued by the firmware for cache synchronization purposes.
8	LMAP. SETV.E	Load map from even register and set valid bit. The LMAP.SETV.E transfer loads a map image descriptor (MID) from the map even data in register into the map RAM and sets the MID's valid bit.
9	LMAP. SETV.O	Load map from odd register and set valid bit. The LMAP.SETV.O transfer loads a map image descriptor (MID) from the map odd data in register into the map RAM and sets the MID's valid bit.
A	WRITE	Write. The WRITE transfer is a CPU data write to main memory or cache. The memory address from the PMAR Bus is sent to the cache and, via the destination bus, to main memory. At the same time, data from the cache data in register is sent to the cache and, via the data bus, to main memory.
B	WRITE. UNLOCK	Write and unlock. The WRITE.UNLOCK transfer is a CPU write to main memory (WRITE) that will unlock a previously locked portion of main memory. A CPU that has implemented a data read and lock on main memory (DREAD.LOCK) must subsequently issue a WRITE.UNLOCK. If the WRITE.UNLOCK transfer is not given, that memory location will be closed to further access, including the CPU that locked it. This prevents the changing of memory by any CPU other than the one that originally issued the lock.

Table 4-24
SelBUS Codes Field CROM 56 through 59 (Sheet 3 of 3)

Value	Syntax	Detailed Definition
C	DREAD. LOCK	Data read and lock. The DREAD.LOCK transfer is a CPU data read from main memory (DREAD) that locks a portion of main memory. No other processor can access that part of memory until the processor that issued the data read and lock also issues a WRITE.UNLOCK transfer (SelBUS code B). The DREAD.LOCK is used in zero bit and set bit macro-instructions.
D	WRITE. N.MAP	Write without map. The WRITE.N.MAP transfer is a CPU write to main memory or cache without use of the map for address calculations. The memory address from the PMAR Bus is sent to the cache and, via the destination bus, to main memory. At the same time, data from the cache data in register is sent to the cache and, via the data bus, to main memory.
E	RSTX	Read status transfer. The RSTX is used in conjunction with the ARSTX (SelBUS overlay code D) to request a status transfer from an I/O channel. After the I/O channel has been preconditioned by the advance read status transfer (ARSTX) and the READY signal returned to the CPU, the RSTX causes the I/O channel to transfer the assembled status to the CPU via the SelBUS.
F	ICT	Interrupt control transfer. The ICT is used in conjunction with the AICT (SelBUS overlay code E) to control the interrupt logic of the SelBUS interface and the microprogram of the I/O device. This transfer is sent from the CPU to the I/O channel during interrupt control instructions and interrupt sequences. After the AICT has preconditioned the I/O device's firmware, and a READY signal has been returned to the CPU, the ICT causes the I/O firmware to execute the interrupt control instruction and send a data return transfer (DRT) to the CPU. The DRT acknowledges the ICT.

Table 4-25
SelBUS Overlay CROM 56 through 59 (Sheet 1 of 3)

Reference: IE Unit logic drawing 130-103584, sheet 11

General: The SelBUS overlay replaces the SelBUS codes field, table 4-24, when enabled by the TTYPE.OVL order (group 0, line 2).

Value	Syntax	Detailed Definition
0	CLR. CACHE.I.0	Clear instruction cache bank 0. The CLR.CACHE.I.0 transfer resets the valid bit of one memory location in bank 0 of the instruction cache. The memory address is provided by the EMAR register.
1	CLR. CACHE.I.1	Clear instruction cache bank 1. The CLR.CACHE.I.1 transfer resets the valid bit of one memory location in bank 1 of the instruction cache. The memory address is provided by the EMAR register.
2	LMAP. HIT.RAM	Load map hit RAM. The LMAP.HIT.RAM transfer causes the 16 bits from the map hit register to be loaded into the map hit RAM. The RAM address is provided by either the EMAR or the LMAR.
3		Not Used.
4	DREAD. N.MAP	Data read without map. The DREAD.N.MAP transfer is a CPU data read from main memory or cache (SelBUS code 6) without use of the map for address calculations. The memory address is obtained from the EMAR register and the result of the data read is fed to the cache data out even (MSW) register.
5	DRT.WRITE	Write returned data from memory into cache. The DRT.WRITE transfer is a hardware code issued by the CS unit. When, during a data read transfer (DREAD), the addressed memory location is not found in cache, the data is read from the same location in main memory. The returned data is sent to one of the cache data out registers, the I2 indirect register, or the I3 registers. At the same time, the CS unit gives the DRT.WRITE transfer, updating the cache with the new memory location and data.
6	INVALIDATE. CACHE	Invalidate cache location. The INVALIDATE.CACHE transfer invalidates a single cache memory location. When a main memory location is modified by an I/O controller, this transfer is used by the CS unit hardware to invalidate the corresponding location in cache.

Table 4-25
SelBUS Overlay CROM 56 through 59 (Sheet 2 of 3)

Value	Syntax	Detailed Definition
7	CACHE. DINTOI3	Cache data in to I3. The CACHE.DINTOI3 transfer is used by the firmware to insert data into the instruction pipeline. Data is fed, via the CAMUX bus, from the cache data in register to the I3 registers in the IE and MS units.
8	CLR.CACHE. D.0	Clear data cache bank 0. The CLR.CACHE.D.O transfer resets the valid bit of one memory location in bank 0 of the data cache. The memory address is provided by the EMAR register.
9	CLR.CACHE. D.1	Clear data cache bank 1. The CLR.CACHE.D.1 transfer resets the valid bit of one memory location in bank 1 of the data cache. The memory address is provided by the EMAR register.
A	READ.I. CACHE	Read from instruction cache. The READ.I.CACHE transfer enables the firmware to read directly from the instruction cache without using the SelBUS. This transfer is used for microdiagnostic purposes.
B	WRITE.I. CACHE	Write to instruction cache. The WRITE.I.CACHE transfer enables the firmware to write directly to the instruction cache without using the SelBUS. This transfer is used for microdiagnostic purposes.
C	WDOT	Write data or order transfer. The WDOT is used in conjunction with the AWDOT (SelBUS overlay code F) to initiate or halt operations within an I/O channel. After the advance write data or order transfer (AWDOT) has determined that the I/O channel is operating and available, the WDOT is sent to the I/O channel. The channel is addressed by destination bus bits 09 through 15. The individual peripheral device is specified by the subaddress field of the WDOT destination bus. The I/O channel does not execute a bus transfer response to the WDOT.
D	ARSTX	Advance read status transfer. The ARSTX is used in conjunction with the RSTX (SelBUS code E) to request a status transfer from an I/O channel to the CPU. The ARSTX is sent from the CPU to the selected I/O channel. This preconditions the I/O channel for a status transfer and causes the requested status to be assembled. When the requested status is assembled and ready for transfer to the CPU, the I/O channel issues a READY signal to the CPU via the SelBUS. The CPU responds to the READY signal by sending a read status transfer (RSTX) to the I/O channel. All advance transfers clear the ready flip-flop in the CS unit.

Table 4-25
SelBUS Overlay CROM 56 through 59 (Sheet 3 of 3)

Value	Syntax	Detailed Definition
E	AICT	Advance interrupt control transfer. The AICT is used in conjunction with the ICT (SelBUS code F) to control the interrupt logic of the SelBUS interface and the I/O device's firmware. The AICT originates in the CPU and preconditions the microprogram of the I/O device for interrupt control action. The I/O device's firmware performs the setup procedures required before the interrupt control action can be executed. When the I/O device is ready to execute the interrupt control instruction, it generates a READY signal to the CPU via the SelBUS. The CPU responds to the READY signal by sending an interrupt control transfer (ICT) to the I/O channel. All advance transfers clear the ready flip-flop in the CS unit.
F	AWDOT	Advance write data or order transfer. The AWDOT is used in conjunction with the WDOT (SelBUS overlay code C) to initiate or halt operations within an I/O channel. The AWDOT is sent from the CPU to the I/O channel to assure that the channel is both operating and available. If these conditions are met, a READY signal is sent from the I/O channel to the CPU. The CPU responds to the READY signal by issuing a write data or order transfer WDOT to the I/O channel. All advance transfers clear the ready flip-flop in the CS unit.

INDEX

ACS mode, 2-55, 2-56
adder, 3-way, 3-36
address logic, effective, 3-62
address selection multiplexer, 3-16
address specification errors, 3-80
address stop commands, 2-76, 2-77
addressing option, 1-7
alterable control store (ACS), 3-29
ALU function field, 4-7, 4-37(T)
ALU shift code field, 4-10, 4-70(T)
arithmetic logic unit (ALU), 3-53, 3-45(T)
attention command, 2-75

backdate program counter, 3-35
base driver, 3-36
base register, 3-35
branch address field, 4-9
byte constant, 3-36
byte constant field, 4-9

cache, 3-60, 3-80, 3-81(F)
    addressing, 3-83
    comparators, 3-61
    control register, 3-85
    CPU write to cache, 3-84
    data in register, 3-61
    data out logic, 3-1
    data out registers, 3-15
    externally originated writes, 3-84
    index RAMs, 3-61
    instruction store, 2-60
    memory logic, 3-13, 3-64
    multiplexer, 3-61
    organization, 3-83
    read from cache, 3-84
    software control, 2-57, 2-58
    system reset, 2-60
    transactions, IE unit, 3-45(T)
    transfer type logic, 3-44
    write to cache, 3-83
cache SelBUS (CS) unit, 1-1, 3-2, 3-11(F), 3-56, 3-57(F)
cache SelBUS (CS) unit components, 3-59
cache SelBUS (CS) unit operation, 3-62
CAMUX bus, 3-32
CAMUX bus functions, 3-43
CC select field, 4-9
central processing unit (CPU), 1-1, 3-3(F), 3-5(F)
clock generation (MS), 3-13

clock generator, 3-38
CMC instruction, 2-59
condition code logic, 3-37, 3-39, 3-42(T)
constant, 32-bit, 3-36
control register (CREG), 3-38, 3-39
control store, 3-28
control store logic, 3-37
CPU overview, 3-1
CROM bus, 3-7
CROM bus functions, 3-38
CS unit, 1-1, 3-2, 3-59, 3-60, 3-62, 3-80

data return transfer (DRT), 3-61
DB bus, 3-15
DB bus functions, 3-50
DB bus source decode, 3-37
decode exceptions, 3-31
decode save, 3-16
diagnostic tape/disc format, 2-39(F)
diagnostics, 1-8

effective address logic, 3-62
EMAR, 3-59
external destination field, 4-8, 4-39(T)
external source field, 4-7, 4-28(T)

floating-point accelerator (FPA), 2-68
FPA, software control, 2-68

general purpose registers, 3-35

halt command, 2-72
halt indicator, 2-70
hop address field, 4-9

I0 register, 3-37
I1 register, 3-35
I1/I0 status register, 3-35
I2 bus, 3-35
I2 bus functions, 3-49
I2 indirect register, 3-32
I2 register, 3-32
I3 and I2 MS copy registers, 3-15
I3 PC register, 3-60
I3 right halfword register, 3-32
IE unit, 1-1, 3-1, 3-32, 3-38
index MAR buffers, 3-61
index register, 3-35
indicators, state, 2-74(T)
initial program load (IPL), 2-10, 2-11, 2-25
    abort, auto IPL, 2-10
    command, 2-72
    data transfers, 2-35
    device address, 2-26
    fault isolation, auto IPL, 2-12

  identify I/O protocol, 2-32
  IOCDs, 2-32
  jumpers, auto IPL, 2-1
  limitations, auto IPL, 2-9
  manual, 2-11
  start I/O, 2-35
  termination, 2-39
initialization, CPU/IPU, 2-4
initialization, system, 2-2
input source multiplexer, 3-60
input/output processor (IOP), 1-3
instruction attribute decode, 3-71(T), 3-74(T)
instruction attributes, 2-79, 2-8(T), 2-94(T)
instruction cache register, 3-32
instruction decode, 3-15
instruction execution (IE) unit, 1-1, 3-1, 3-9(F), 3-32, 3-33(F), 3-38
instruction execution (IE) unit components, 3-32
instruction/execution (IE) unit operation, 3-38
instruction/operand stop, 3-86
integrated memory module (IMM), 1-3
internal destination field, 4-8, 4-38(T)
internal processing unit (IPU), 1-5
  operation, 2-43
  operational characteristics, 2-44
  software initialization, 2-10
internal source field, 4-7, 4-36(T)
interprocessor bit semaphores, 2-61, 2-64(T)
interrupt indicator, 2-71
I/O class E, 2-34(T)
I/O class F, 2-33(T), 2-34(T)
I/O classes, CPU and IPU, 2-45(T)
I/O expansion chassis, 1-3, 1-4(F)
IOP panel address stop, 2-60
I/O transfers, 1-8
IPU, 1-5, 2-10, 2-43, 2-44
  IPU console, 2-46
  channel address, 2-47
  channel interrupt level, 2-48
  channel subaddress, 2-48
  Class 6 I/O, 2-49
  Class 7 I/O, 2-49
  Class B I/O, 2-49
  configuration, 2-47
  external interrupt levels, 2-49
  interval timer device address, 2-48
  interval timer interrupt level, 2-48
  I/O class identification, 2-49
  physical address, 2-47
  real-time clock interrupt level, 2-48

jumpers, auto IPL, 2-1
jumpers, system, 2-1

L-branch address field, 4-9
L-order select field, 4-9

- leading trailing zeros detect, 3-31
- leap address field, 4-9
- least recently used (L.R.U) circuitry, 3-84
- left shifters, 3-61
- level orders, 3-38, 3-39
- literal/scratch address field, 4-9
- LMAR copy, 3-36
- LMAR counter, 3-60
- logical address logic, 3-13
- look ahead multiplexer, I1/I0, 3-37

- machine cycles, 3-13
- macroprogram counter, 3-60
- maintainability features, 1-8
- map, memory, 3-59, 3-88, 3-89(F)
  - bypass multiplexer, 3-60, 3-93
  - current process index (CPIX), 3-88
  - data in register, 3-62
  - data structures, 3-88, 3-90(F)
  - entries, 3-92
  - hit RAM, 3-60
  - initialization, 3-91
  - load, 3-92
  - logical to physical address conversion, 3-93
  - look aside buffer, 3-92
  - map image descriptor (MID), 3-91
  - map image descriptor list (MIDL), 3-91
  - master process list (MPL), 3-88
  - miss MAR, 3-36
  - RAM, 3-60
- MAR comparator, 3-60
- MAR polling logic, 3-43
- memory environments, 1-6
- memory address register, effective, 3-36
- memory address virtual command, 2-78
- memory management logic, 3-13, 3-63
- microinstruction decode, 3-51
- microinterrupt control, 3-16
- microinterrupt logic, 3-44
- microinterrupt priority levels 3-47(T)
- microprocessor 2901, 3-37, 3-50, 3-51(F), 3-77(F)
  - data outputs, 3-53
  - division routine, 3-53
  - multiplication routine, 3-53
  - Q register, 3-53
  - RAM, 3-53
  - shifting, 3-51
- microprogram counter, 3-16, 3-13(F)
- microprogram counter latch, 3-19
- microprogram counter output buffer, 3-19
- microprogram counter save register, 3-16
- microsequencer (MS) components, 3-15
- microsequencer (MS) unit, 1-1, 3-1, 3-7(F), 3-13, 3-17(F)
- microword, 4-1, 4-3(F)
- microword field descriptions, 4-5

MPX-32 tape format, 2-38(F)
MS unit 1-1, 3-1, 3-13, 3-15

nucleus, computer, 1-1, 1-2(F)

operand/transaction request logic, 3-44
operating mode, 1-5
operator command, 2-72
operator console, 1-5
operator indicators, 2-70
options, software control of, 2-52
order enable field, 4-8, 4-48(T)
order group field, 4-8, 4-66(T)
order structure, 3-30
order structure, 3-38

parity error indicator, 2-71
P, C, and H fields, 4-9
pipeline, 3-2
pipeline, CPU fill, 2-38
pipeline, advance routine, 3-76
pipeline, auto microinterrupt, 3-80
pipeline, backing store stage, 3-70
pipeline, conflict logic, 3-79
pipeline, conflict logic, 3-79
pipeline, decode stage, 3-72
pipeline, execution stage, 3-76
pipeline, instruction, 3-68, 3-69(F)
pipeline, vector stage, 3-73
PMAR, 3-61
PMAR copy register, 3-62
powerdown sequence, 2-39
powerdown sequencing flowchard, 2-43(F)
powerup automatic trap halt, 2-10, 2-11(F)
powerup, auto IPL, 2-9
powerup, auto restart, 2-4, 2-5(F)
predecoder, 3-32
prefetch, 2-60
privileged state, 1-6
processor select switch, 2-1
program counter, micro, 3-1, 3-16, 3-23(F)
program status word register, 3-37
PROM mode, 2-55

read effective address command, 2-78
register A field, 4-6, 4-24(T)
register B field, 4-7, 4-26(T)
reliability features, 1-8
reset, 2-11, 2-15(F)
reset command, 2-72
reset, firmware, 2-26
right halfword, register, 3-32
right shifter, 3-15
run command, 2-72
run indicator, 2-71

scratchpad, CPU/IPU, 2-50
    allocation of, 2-51(F)
    device entry , 2-52, 2-53(F)
    image, 2-50
    interrupt entry, 2-52, 2-54(F)
    keyword, 2-52
    logic, 3-1
SelBUS codes field, 4-10, 4-72(T)
SelBUS data in register, 3-62
SelBUS data out register, 3-61
SelBUS interface, 3-66
SelBUS interface logic, 3-2
SelBUS PMART, 3-62
semaphores, interprocessor bit, 2-61, 2-64(T)
sequence field, 4-5, 4-10(T)
shared memory, 2-60, 2-61(F), 2-62(F)
shared memory detect, 3-61, 3-86
shifter (F and C bits), 3-85
sign extend, 3-36
SMC instruction, 2-59
SMC read and lock control, 2-65, 2-66(T)
SMC shared range control, 2-65
software, 1-8
specifications, 1-9, 1-10(T)
state indicators, 2-74(T)
step command, 2-77
stopclock logic, 3-54, 3-55(T)
system control panel commands, 2-73(T)

test field, 4-6, 4-12(T)
test structure, 3-37, 3-39, 3-40(F), 3-41(F)
timing, basic diagram, 3-14(F)
trap halt, automatic, 2-49
trap vector locations, 2-45(T)
turnkey panel, 1-5, 1-6(F), 3-2
turnkey panel interface logic, 3-31

wait indicator, 2-71
writable control store (WCS), 1-1, 2-67, 3-29, 3-87
    address register, 3-19
    entry and use, 3-87
    RAM data registers, 3-19
    software initialization and maintenance, 2-67
    user notes, 3-87
write PC command, 2-75
write PSW command, 2-75
write protection, memory, 1-7
write PSD2 command, 2-75

Y bus, 3-15, 3-37
Y bus control, 3-37
Y bus functions, 3-50

Gould Inc., Computer Systems Division
6901 W. Sunrise Blvd.
P. O. Box 409148
Fort Lauderdale, FL 33340-9148
Telephone (305) 587-2900

GOULD
Electronics

Users Group Membership Application

USER ORGANIZATION: ______________________________

REPRESENTATIVE(S): ______________________________

____

____

ADDRESS: ______________________________

____

TELEX NUMBER: ______________ PHONE NUMBER: ______________

NUMBER AND TYPE OF GOULD CSD COMPUTERS: ______________________________

____

____

____

APPLICATIONS (Please Indicate)

1. EDP
    - A. Inventory Control
    - B. Engineering & Production Data Control
    - C. Large Machine Off-Load
    - D. Remote Batch Terminal
    - E. Other
2. Communications
    - A. Telephone System Monitoring
    - B. Front End Processors
    - C. Message Switching
    - D. Other
3. Design & Drafting
    - A. Electrical
    - B. Mechanical
    - C. Architectural
    - D. Cartography
    - E. Image Processing
    - F. Other
4. Industrial Automation
    - A. Continuous Process Control Op.
    - B. Production Scheduling & Control
    - C. Process Planning
    - D. Numerical Control
    - E. Other
5. Laboratory and Computational
    - A. Seismic
    - B. Scientific Calculation
    - C. Experiment Monitoring
    - D. Mathematical Modeling
    - E. Signal Processing
    - F. Other
6. Energy Monitoring & Control
    - A. Power Generation
    - B. Power Distribution
    - C. Environmental Control
    - D. Meter Monitoring
    - E. Other
7. Simulation
    - A. Flight Simulators
    - B. Power Plant Simulators
    - C. Electronic Warfare
    - D. Other
8. Other

Please return to:

Users Group Administrator

Date: ______________

Gould Inc., Computer Systems Division Users Group. . .

The purpose of the Gould CSD Users Group is to help create better User/User and User/Gould CSD Communications.

There is no fee to join the Users Group. Simply complete the Membership Application on the reverse side and mail to the Users Group Administrator. You will automatically receive Users Group Newsletters, Referral Guide and other pertinent Users Group Activity information.

Fold and Staple for Mailing

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 947 FT. LAUDERDALE, FLORIDA

POSTAGE WILL BE PAID BY ADDRESSEE

GOULD INC., COMPUTER SYSTEMS DIVISION
ATTENTION: USERS GROUP ADMINISTRATOR
6901 W. SUNRISE BLVD.
P. O. BOX 409148
FT. LAUDERDALE FL 33340-9970

(Detach Here)

Fold and Staple for Mailing

GOULD
Electronics

MP MASTER
MA34644